The Stata Journal (*yyyy*) *vv*, Number *ii*

lassopack: Model selection and prediction with regularized regression in Stata

Achim Ahrens
The Economic and Social Research Institute
Dublin, Ireland
achim.ahrens@esri.ie

Christian B. Hansen
University of Chicago
christian.hansen@chicagobooth.edu

Mark E. Schaffer
Heriot-Watt University
Edinburgh, United Kingdom
m.e.schaffer@hw.ac.uk

Abstract. This article introduces **lassopack**, a suite of programs for regularized regression in Stata. **lassopack** implements lasso, square-root lasso, elastic net, ridge regression, adaptive lasso and post-estimation OLS. The methods are suitable for the high-dimensional setting where the number of predictors p may be large and possibly greater than the number of observations, n. We offer three different approaches for selecting the penalization ('tuning') parameters: information criteria (implemented in `lasso2`), K-fold cross-validation and h-step ahead rolling cross-validation for cross-section, panel and time-series data (`cvlasso`), and theory-driven ('rigorous' or plug-in) penalization for the lasso and square-root lasso for cross-section and panel data (`rlasso`). We discuss the theoretical framework and practical considerations for each approach. We also present Monte Carlo results to compare the performance of the penalization approaches.

Keywords: st0001, lasso2, cvlasso, rlasso, lasso, elastic net, square-root lasso, cross-validation

1 Introduction

Machine learning is attracting increasing attention across a wide range of scientific disciplines. Recent surveys explore how machine learning methods can be utilized in economics and applied econometrics (Varian 2014; Mullainathan and Spiess 2017; Athey 2017; Kleinberg et al. 2018; Athey and Imbens 2019). At the same time, Stata offers to date only a limited set of machine learning tools. **lassopack** is an attempt to fill this gap by providing easy-to-use and flexible methods for regularized regression in Stata.[1]

While regularized linear regression is only one of many methods in the toolbox of machine learning, it has some properties that make it attractive for empirical research. To begin with, it is a straightforward extension of linear regression. Just like ordinary least squares (OLS), regularized linear regression minimizes the sum of squared devia-

1. This article refers to version 1.2 of **lassopack** released on the 15th of January, 2019. For additional information and data files, see https://statalasso.github.io/. As of version 1.3, **lassopack** also supports logistic lasso regression which is not covered in this article; see `help lassologit` for more information.

tions between observed and model predicted values, but imposes a regularization penalty aimed at limiting model complexity. The most popular regularized regression method is the lasso—which this package is named after—introduced by Frank and Friedman (1993) and Tibshirani (1996), which penalizes the absolute size of coefficient estimates.

The primary purpose of regularized regression, like supervised machine learning methods more generally, is prediction. Regularized regression typically does not produce estimates that can be interpreted as causal and statistical inference on these coefficients is complicated.[2] While regularized regression may select the true model as the sample size increases, this is generally only the case under strong assumptions. However, regularized regression can aid causal inference without relying on the strong assumptions required for perfect model selection. The post-double-selection methodology of Belloni et al. (2014a) and the post-regularization approach of Chernozhukov et al. (2015) can be used to select appropriate control variables from a large set of putative confounding factors and, thereby, improve robustness of estimation of the parameters of interest. Likewise, the first stage of two-step least-squares is a prediction problem and lasso or ridge can be applied to obtain optimal instruments (Belloni et al. 2012; Carrasco 2012; Hansen and Kozbur 2014). These methods are implemented in our sister package **pdslasso** (Ahrens et al. 2018), which builds on the algorithms developed in **lassopack**.

The strength of regularized regression as a prediction technique stems from the *bias-variance trade-off*. The prediction error can be decomposed into the unknown error variance reflecting the overall noise level (which is irreducible), the squared estimation bias and the variance of the predictor. The variance of the estimated predictor is increasing in the model complexity, whereas the bias tends to decrease with model complexity. By reducing model complexity and inducing a shrinkage bias, regularized regression methods tend to outperform OLS in terms of *out-of-sample* prediction performance. In doing so, regularized regression addresses the common problem of overfitting: high in-sample fit (high R^2), but poor prediction performance on unseen data.

Another advantage is that the regularization methods of **lassopack**—with the exception of ridge regression—set some coefficients to exactly zero, thereby excluding predictors from the model. In this way, regularized regression can serve as a model selection technique. Especially when faced with a large number of putative predictors, model selection is challenging. Iterative testing procedures, such as the *general-to-specific approach*, typically induce pre-testing biases and hypothesis tests often lead to many false positives. At the same time, high-dimensional problems where the number of predictors is large relative to the sample size are a common phenomenon, especially when the true model is treated as unknown. Regularized regression allows for identification even when the number of predictors exceeds the sample size under the *sparsity assumption* which requires that the true model only includes a small number of predictors or, more generally, that only few predictors are necessary to approximate the true model sufficiently well.

Regularized regression methods rely on tuning parameters that control the degree

2. This is an active area of research, see for example Buhlmann (2013); Meinshausen et al. (2009); Weilenmann et al. (2017); Wasserman and Roeder (2009); Lockhart et al. (2014).

and type of penalization. **lassopack** offers three approaches to select these tuning parameters. The classical approach is to select tuning parameters using cross-validation in order to optimize *out-of-sample* prediction performance. Cross-validation methods are universally applicable and generally perform well for prediction tasks, but are computationally expensive. A second approach relies on information criteria such as the *Akaike information criterion* (Zou et al. 2007; Zhang et al. 2010). Information criteria are easy to calculate and have attractive theoretical properties, but are less robust to violations of the independence and homoskedasticity assumptions (Arlot and Celisse 2010). Rigorous penalization for the lasso and square-root lasso offers a third option. The tuning parameters are chosen to theoretically provide good prediction performance and control overfitting in the presence of heteroskedastic, non-Gaussian and cluster-dependent errors (Belloni et al. 2012, 2014b, 2016). The rigorous approach places a high priority on controlling overfitting, thus often producing parsimonious models. This strong focus on containing overfitting is of practical and theoretical benefit for selecting control variables or instruments in a structural model, but also implies that the approach may be outperformed by cross-validation techniques for pure prediction tasks. Which approach is most appropriate depends on the type of data at hand and the purpose of the analysis. To provide guidance for applied reseachers, we discuss the theoretical foundation of all three approaches, and present Monte Carlo results that assess their relative performance.

The article proceeds as follows. In Section 2, we present the estimation methods implemented in **lassopack**. Section 3-5 discuss the aforementioned approaches for selecting the tuning parameters: information criteria in Section 3, cross-validation in Section 4 and rigorous penalization in Section 5. The three commands, which correspond to the three penalization approaches, are presented in Section 6, followed by demonstrations in Section 7. Section 8 presents Monte Carlo results. Further technical notes are in Section 9.

Notation. We briefly clarify the notation used in this article. Suppose $\boldsymbol{a}$ is a vector of dimension m with typical element a_j for $j = 1, \ldots, m$. The ℓ_1-norm is defined as $\|\boldsymbol{a}\|_1 = \sum_{j=1}^{m} |a_j|$, and the ℓ_2-norm is $\|\boldsymbol{a}\|_2 = \sqrt{\sum_{j=1}^{m} |a_j|^2}$. The '$\ell_0$-norm' of $\boldsymbol{a}$ is denoted by $\|a\|_0$ and is equal to the number of non-zero elements in $\boldsymbol{a}$. $\mathbb{1}\{.\}$ denotes the indicator function. We use the notation $b \vee c$ to denote the maximum value of b and c, i.e., $\max(b, c)$.

2 Regularized regression

This section introduces the regularized regression methods implemented in **lassopack**. We consider the high-dimensional linear model

$$y_i = \boldsymbol{x}_i'\boldsymbol{\beta} + \varepsilon_i, \qquad i = 1, \ldots, n,$$

where the number of predictors, p, may be large and even exceed the sample size, n. The regularization methods introduced in this section can accommodate large-p models

under the assumption of sparsity: out of the p predictors only a subset of $s \ll n$ are included in the true model where s is the sparsity index

$$s := \sum_{j=1}^{p} \mathbb{1}\{\beta_j \neq 0\} = \|\boldsymbol{\beta}\|_0 .$$

We refer to this assumption as *exact sparsity*. It is more restrictive than required, but we use it here for illustrative purposes. We will later relax the assumption to allow for non-zero, but 'small', β_j coefficients. We also define the active set $\Omega = \{j \in \{1, \ldots, p\} : \beta_j \neq 0\}$, which is the set of non-zero coefficients. In general, p, s, Ω and $\boldsymbol{\beta}$ may depend on n but we suppress the n-subscript for notational convenience.

We adopt the following convention throughout the article: unless otherwise noted, all variables have been mean-centered such that $\sum_i y_i = 0$ and $\sum_i x_{ij} = 0$, and all variables are measured in their natural units, i.e., they have *not* been pre-standardized to have unit variance. By assuming the data have already been mean-centered we simplify the notation and exposition. Leaving the data in natural units, on the other hand, allows us to discuss standardization in the context of penalization.

Penalized regression methods rely on tuning parameters that control the degree and type of penalization. The estimation methods implemented in **lassopack**, which we will introduce in the following sub-section, use two tuning parameters: λ controls the general degree of penalization. Setting λ to zero corresponds to no penalization and is equivalent to OLS, while a sufficiently large value of λ yields an empty model, where all coefficients are zero. The second tuning parameter, α, determines the relative contribution of ℓ_1 (lasso-type) vs. ℓ_2 (ridge-type) penalization. We will discuss the merits and properties of ℓ_1 and ℓ_2 penalization in the following sub-section. The three approaches offered by **lassopack** for selecting λ and α are introduced in 2.2.

2.1 The estimators

Lasso

The lasso takes a special position, as it provides the basis for the rigorous penalization approach (see Section 5) and has inspired other methods such as elastic net and square-root lasso, which are introduced later in this section. The lasso minimizes the mean squared error subject to a penalty on the absolute size of coefficient estimates:

$$\hat{\boldsymbol{\beta}}_{\text{lasso}}(\lambda) = \arg\min \frac{1}{n} \sum_{i=1}^{n} (y_i - \boldsymbol{x}_i'\boldsymbol{\beta})^2 + \frac{\lambda}{n} \sum_{j=1}^{p} \psi_j |\beta_j|. \tag{1}$$

The tuning parameter λ controls the overall penalty level and ψ_j are predictor-specific penalty loadings.

Tibshirani (1996) motivates the lasso with two major advantages over OLS. First, due to the nature of the ℓ_1-penalty, the lasso sets some of the coefficient estimates exactly to zero and, in doing so, removes some predictors from the model. Thus, the lasso serves

as a model selection technique and facilitates model interpretation. Secondly, lasso can outperform least squares in terms of prediction accuracy due to the bias-variance trade-off.

The lasso coefficient path, which constitutes the trajectory of coefficient estimates as a function of λ, is piecewise linear with changes in slope where variables enter or leave the active set. The change points are referred to as *knots*.

The lasso, unlike OLS, is not invariant to linear transformations, which is why scaling matters. If the predictors are not of equal variance, the most common approach is to pre-standardize the data such that $\frac{1}{n}\sum_i x_{ij}^2 = 1$ and set $\psi_j = 1$ for $j = 1, \dots, p$. Alternatively, we can set the penalty loadings to $\hat{\psi}_j = (\frac{1}{n}\sum_i x_{ij}^2)^{-1/2}$. The two methods yield identical results.

Ridge regression

Ridge regression (Tikhonov 1963; Hoerl and Kennard 1970) replaces the ℓ_1-penalty of the lasso with a ℓ_2-penalty, thus minimizing

$$\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \boldsymbol{x}_i'\boldsymbol{\beta}\right)^2 + \frac{\lambda}{n}\sum_{j=1}^{p}\psi_j^2\beta_j^2. \tag{2}$$

The interpretation and choice of the penalty loadings ψ_j is the same as above. As in the case of the lasso, we need to account for uneven variance, either through pre-estimation standardization or by appropriately choosing the penalty loadings ψ_j.

In contrast to estimators relying on ℓ_1-penalization, the ridge does not perform variable selection. At the same time, it also does not rely on the assumption of sparsity. This makes the ridge attractive in the presence of dense signals, i.e., when the assumption of sparsity does not seem plausible. Dense high-dimensional problems are more challenging than sparse problems: for example, Dicker (2016) shows that, if $p/n \to \infty$, it is not possible to outperform a trivial estimator that only includes the constant. If $p, n \to$ jointly, but p/n converges to a finite constant, the ridge has desirable properties in dense models and tends to perform better than sparsity-based methods (Hsu et al. 2014; Dicker 2016; Dobriban and Wager 2018).

Ridge regression is closely linked to principal component regression. Both methods are popular in the context of multicollinearity due to their low variance relative to OLS. Principal components regression applies OLS to a subset of components derived from principal component analysis; thereby discarding a specified number of components with low variance. The rationale for removing low-variance components is that the predictive power of each component tends to increase with the variance. The ridge can be interpreted as projecting the response against principal components while imposing a higher penalty on components exhibiting low variance. Hence, the ridge follows a similar principle; but, rather than discarding low-variance components, it applies a more severe shrinkage (Hastie et al. 2009).

A comparison of lasso and ridge regression provides further insights into the nature

of ℓ_1 and ℓ_2 penalization. For this purpose, it is helpful to write lasso and ridge in constrained form as

$$\hat{\boldsymbol{\beta}}_{\text{lasso}} = \arg\min \frac{1}{n}\sum_{i=1}^{p}\left(y_i - \boldsymbol{x}_i'\boldsymbol{\beta}\right)^2 \quad \text{subject to} \quad \sum_{j=1}^{n}\psi_j|\beta_j| \leq \tau,$$

$$\hat{\boldsymbol{\beta}}_{\text{ridge}} = \arg\min \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \boldsymbol{x}_i'\boldsymbol{\beta}\right)^2 \quad \text{subject to} \quad \sum_{j=1}^{p}\psi_j^2\beta_j^2 \leq \tau$$

and to examine the shapes of the constraint sets. The above optimization problems use the tuning parameter τ instead of λ. Note that there exists a data-dependent relationship between λ and τ.

Figure 1 illustrates the geometry underpinning lasso and ridge regression for the case of $p = 2$ and $\psi_1 = \psi_2 = 1$ (i.e., unity penalty loadings). The grey elliptical lines represent residual sum of squares contours and the black lines indicate the lasso and ridge constraints. The lasso constraint set, given by $|\beta_1| + |\beta_2| \leq \tau$, is diamond-shaped with vertices along the axes from which it immediately follows that the lasso solution may set coefficients exactly to 0. In contrast, the ridge constraint set, $\beta_1^2 + \beta_2^2 \leq \tau$, is circular and will thus (effectively) never produce a solution with any coefficient set to 0. Finally, $\hat{\beta}_0$ in the figure denotes the solution without penalization, which corresponds to OLS. The lasso solution at the corner of the diamond implies that, in this example, one of the coefficients is set to zero, whereas ridge and OLS produce non-zero estimates for both coefficients.

While there exists no closed form solution for the lasso, the ridge solution can be expressed as

$$\hat{\boldsymbol{\beta}}_{\text{ridge}} = (\boldsymbol{X}'\boldsymbol{X} + \lambda\boldsymbol{\Psi}'\boldsymbol{\Psi})^{-1}\boldsymbol{X}'\boldsymbol{y}.$$

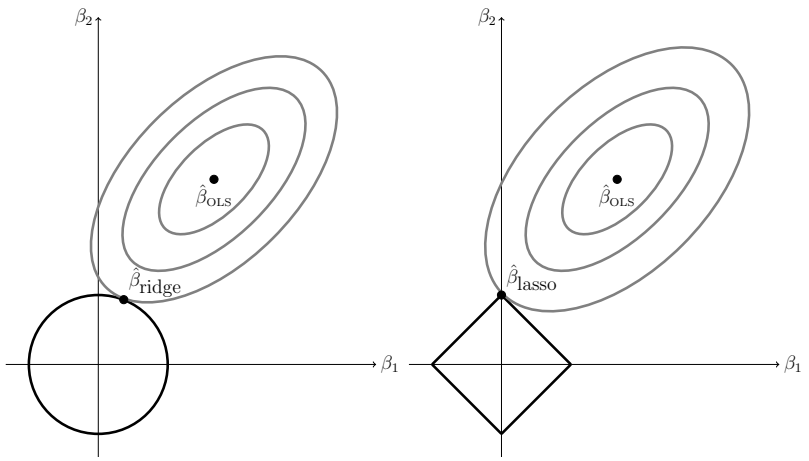
Here $\boldsymbol{X}$ is the $n \times p$ matrix of predictors with typical element x_{ij}, $\boldsymbol{y}$ is the response vector and $\boldsymbol{\Psi} = \text{diag}(\psi_1, \ldots, \psi_p)$ is the diagonal matrix of penalty loadings. The ridge regularizes the regressor matrix by adding positive constants to the diagonal of $\boldsymbol{X}'\boldsymbol{X}$. The ridge solution is thus well-defined generally as long as all the ψ_j and λ are sufficiently large even if $\boldsymbol{X}'\boldsymbol{X}$ is rank-deficient.

Elastic net

The elastic net of Zou and Hastie (2005) combines some of the strengths of lasso and ridge regression. It applies a mix of ℓ_1 (lasso-type) and ℓ_2 (ridge-type) penalization:

$$\hat{\boldsymbol{\beta}}_{\text{elastic}} = \arg\min \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \boldsymbol{x}_i'\boldsymbol{\beta}\right)^2 + \frac{\lambda}{n}\left[\alpha\sum_{j=1}^{p}\psi_j|\beta_j| + (1-\alpha)\sum_{j=1}^{p}\psi_j^2\beta_j^2\right] \qquad (3)$$

The additional parameter α determines the relative to contribution of ℓ_1 vs. ℓ_2 penalization. In the presence of groups of correlated regressors, the lasso typically selects only one variable from each group, whereas the ridge tends to produce similar coefficient estimates for groups of correlated variables. On the other hand, the ridge does not



(a) Ridge (b) Lasso

Figure 1: Behaviour of ℓ_1 and ℓ_2-penalty in comparison. Gray lines represent RSS contour lines and the black lines represent the lasso and ridge constraint, respectively. $\hat{\beta}_{\text{OLS}}$ denotes the OLS estimate. $\hat{\beta}_{\text{lasso}}$ and $\hat{\beta}_{\text{ridge}}$ are the lasso and ridge estimate. The illustration is based on Tibshirani, 1996, Fig. 2.

yield sparse solutions impeding model interpretation. The elastic net is able to produce sparse solutions for some α greater than zero, and retains or drops correlated variables jointly.

Adaptive lasso

The *irrepresentable condition (IRC)* is shown to be sufficient and (almost) necessary for the lasso to be model selection consistent (Zhao and Yu 2006; Meinshausen and Bühlmann 2006). However, the IRC imposes strict constraints on the degree of correlation between predictors in the true model and predictors outside of the model. Motivated by this non-trivial condition for the lasso to be variable selection consistent, Zou (2006) proposed the adaptive lasso. The adaptive lasso uses penalty loadings of $\psi_j = 1/|\hat{\beta}_{0,j}|^\theta$ where $\hat{\beta}_{0,j}$ is an initial estimator. The adaptive lasso is variable-selection consistent for fixed p under weaker assumptions than the standard lasso. If $p < n$, OLS can be used as the initial estimator. Huang et al. (2008) prove variable selection consistency for large p and suggest using univariate OLS if $p > n$. The idea of adaptive penalty loadings can also be applied to elastic net and ridge regression (Zou and Zhang 2009).

Square-root lasso

The square-root lasso,

$$\hat{\boldsymbol{\beta}}_{\sqrt{\text{lasso}}} = \arg\min \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \boldsymbol{x}_i'\boldsymbol{\beta})^2} + \frac{\lambda}{n}\sum_{j=1}^{p}\psi_j|\beta_j|, \tag{4}$$

is a modification of the lasso that minimizes the root mean squared error, while also imposing an ℓ_1-penalty. The main advantage of the square-root lasso over the standard lasso becomes apparent if theoretically grounded, data-driven penalization is used. Specifically, the score vector, and thus the optimal penalty level, is independent of the unknown error variance under homoskedasticity as shown by Belloni et al. (2011), resulting in a simpler procedure for choosing λ (see Section 5).[3]

Post-estimation OLS

Penalized regression methods induce an attenuation bias that can be alleviated by post-estimation OLS, which applies OLS to the variables selected by the first-stage variable selection method, i.e.,

$$\hat{\boldsymbol{\beta}}_{\text{post}} = \arg\min \frac{1}{n}\sum_{i=1}^{n}(y_i - \boldsymbol{x}_i'\boldsymbol{\beta})^2 \qquad \text{subject to} \qquad \beta_j = 0 \;\text{ if }\; \tilde{\beta}_j = 0, \tag{5}$$

where $\tilde{\beta}_j$ is a sparse first-step estimator such as the lasso. Thus, post-estimation OLS treats the first-step estimator as a genuine model selection technique. For the case of the lasso with theory-driven regularization, Belloni and Chernozhukov (2013) have shown that post-estimation OLS, also referred to as post-lasso, achieves the same convergence rates as the lasso and can outperform the lasso in situations where consistent model selection is feasible (see also Belloni et al. 2012). Similar results hold for the square-root lasso (Belloni et al. 2011, 2014b).

2.2 Choice of the tuning parameters

Since coefficient estimates and the set of selected variables depend on λ and α, a central question is how to choose these tuning parameters. Which method is most appropriate depends on the objectives and setting: in particular, the aim of the analysis (prediction or model identification), computational constraints, and if and how the i.i.d. assumption is violated. **lassopack** offers three approaches for selecting the penalty level of λ and α:

3. It can be shown that the square-root lasso is equivalent to the *scaled lasso*. The scaled lasso solves the minimization problem

$$\min_{\beta,\sigma} \frac{1}{2\sigma}\sum_{i=1}^{n}\left(y_i - \boldsymbol{x}_i'\boldsymbol{\beta}\right)^2 + \frac{n\sigma}{2} + \lambda\sum_{j=1}^{p}\psi_j|\beta_j|.$$

See Raninen and Ollila (2017) for further discussion and references, including an extension to the scaled or square-root elastic net.

1. *Information criteria:* The value of λ can be selected using information criteria. `lasso2` implements model selection using four information criteria. We discuss this approach in Section 3.

2. *Cross-validation:* The aim of cross-validation is to optimize the out-of-sample prediction performance. Cross-validation is implemented in `cvlasso`, which allows for cross-validation across both λ and the elastic net parameter α. See Section 4.

3. *Theory-driven ('rigorous'):* Theoretically justified and feasible penalty levels and loadings are available for the lasso and square-root lasso via `rlasso`. The penalization is chosen to dominate the noise of the data-generating process (represented by the score vector), which allows derivation of theoretical results with regard to consistent prediction and parameter estimation. See Section 5.

3 Tuning parameter selection using information criteria

Information criteria are closely related to regularization methods. The classical *Akaike's information criterion* (Akaike 1974, AIC) is defined as $-2\times$log-likelihood$+2p$. Thus, the AIC can be interpreted as penalized likelihood which imposes a penalty on the number of predictors included in the model. This form of penalization, referred to as ℓ_0-penalty, has, however, an important practical disadvantage. In order to find the model with the lowest AIC, we need to estimate all different model specifications. In practice, it is often not feasible to consider the full model space. For example, with only 20 predictors, there are more than 1 million different models.

The advantage of regularized regression is that it provides a data-driven method for reducing model selection to a one-dimensional problem (or two-dimensional problem in the case of the elastic net) where we need to select λ (and α). Theoretical properties of information criteria are well-understood and they are easy to compute once coefficient estimates are obtained. Thus, it seems natural to utilize the strengths of information criteria as model selection procedures to select the penalization level.

Information criteria can be categorized based on two central properties: *loss efficiency* and *model selection consistency*. A model selection procedure is referred to as loss efficient if it yields the smallest averaged squared error attainable by all candidate models. Model selection consistency requires that the true model is selected with probability approaching 1 as $n \to \infty$. Accordingly, which information information criteria is appropriate in a given setting also depends on whether the aim of analysis is prediction or identification of the true model.

We first consider the most popular information criteria, AIC and *Bayesian information criterion* (Schwarz 1978, BIC):

$$\begin{aligned}\text{AIC}(\lambda,\alpha) &= n\log\left(\hat{\sigma}^2(\lambda,\alpha)\right) + 2df(\lambda,\alpha),\\ \text{BIC}(\lambda,\alpha) &= n\log\left(\hat{\sigma}^2(\lambda,\alpha)\right) + df(\lambda,\alpha)\log(n),\end{aligned}$$

where $\hat{\sigma}^2(\lambda,\alpha) = n^{-1}\sum_{i=1}^{n}\hat{\varepsilon}_i^2$ and $\hat{\varepsilon}_i$ are the residuals. $df(\lambda,\alpha)$ is the effective degrees of freedom, which is a measure of model complexity. In the linear regression model,

the degrees of freedom is simply the number of regressors. Zou et al. (2007) show that the number of coefficients estimated to be non-zero, $\hat{s}$, is an unbiased and consistent estimate of $df(\lambda)$ for the lasso ($\alpha = 1$). Tibshirani and Taylor (2012) confirm that the result also holds if $p > n$. More generally, the degrees of freedom of the elastic net can be calculated as the trace of the projection matrix, i.e.,

$$\widehat{df}(\lambda, \alpha) = \text{tr}(\boldsymbol{X}_{\hat{\Omega}}(\boldsymbol{X}_{\hat{\Omega}}'\boldsymbol{X}_{\hat{\Omega}} + \lambda(1-\alpha)\boldsymbol{\Psi})^{-1}\boldsymbol{X}_{\hat{\Omega}}').$$

where $\boldsymbol{X}_{\hat{\Omega}}$ is the $n \times \hat{s}$ matrix of selected regressors. The unbiased estimator of the degrees of freedom provides a justification for using the classical AIC and BIC to select tuning parameters (Zou et al. 2007).

The BIC is known to be model selection consistent if the true model is among the candidate models, whereas AIC is inconsistent. Clearly, the assumption that the true model is among the candidates is strong; even the existence of the 'true model' may be problematic, so that loss efficiency may become a desirable second-best. The AIC is, in contrast to BIC, loss efficient. Yang (2005) shows that the differences between AIC-type information criteria and BIC are fundamental; a consistent model selection method, such as the BIC, cannot be loss efficient, and vice versa. Zhang et al. (2010) confirm this relation in the context of penalized regression.

Both AIC and BIC are not suitable in the large-p-small-n context where they tend to select too many variables (see Monte Carlo simulations in Section 8). It is well known that the AIC is biased in small samples, which motivated the bias-corrected AIC (Sugiura 1978; Hurvich and Tsai 1989),

$$\text{AIC}_c(\lambda, \alpha) = n \log\left(\hat{\sigma}^2(\lambda, \alpha)\right) + 2df(\lambda, \alpha)\frac{n}{n - df(\lambda, \alpha)}.$$

The bias can be severe if df is large relative to n, and thus the AIC_c should be favoured when n is small or with high-dimensional data.

The BIC relies on the assumption that each model has the same prior probability. This assumptions seems reasonable when the researcher has no prior knowledge; yet, it contradicts the principle of parsimony and becomes problematic if p is large. To see why, consider the case where $p = 1000$ (following Chen and Chen 2008): There are 1000 models for which one parameter is non-zero ($s = 1$), while there are $1000 \times 999/2$ models for which $s = 2$. Thus, the prior probability of $s = 2$ is larger than the prior probability of $s = 1$ by a factor of $999/2$. More generally, since the prior probability that $s = j$ is larger than the probability that $s = j - 1$ (up to the point where $j = p/2$), the BIC is likely to over-select variables. To address this shortcoming, Chen and Chen (2008) introduce the Extended BIC, defined as

$$\text{EBIC}_\xi(\lambda, \alpha) = n \log\left(\hat{\sigma}^2(\lambda, \alpha)\right) + df(\lambda, \alpha)\log(n) + 2\xi df(\lambda, \alpha)\log(p),$$

which imposes an additional penalty on the size of the model. The prior distribution is chosen such that the probability of a model with dimension j is inversely proportional to the total number of models for which $s = j$. The additional parameter, $\xi \in [0, 1]$,

controls the size of the additional penalty.[4] Chen and Chen (2008) show in simulation studies that the EBIC_ξ outperforms the traditional BIC, which exhibits a higher false discovery rate when p is large relative to n.

4 Tuning parameter selection using cross-validation

The aim of cross-validation is to directly assess the performance of a model on unseen data. To this end, the data is repeatedly divided into a *training* and a *validation data set*. The models are fit to the training data and the validation data is used to assess the predictive performance. In the context of regularized regression, cross-validation can be used to select the tuning parameters that yield the best performance, e.g., the best *out-of-sample* mean squared prediction error. A wide range of methods for cross-validation are available. For an extensive review, we recommend Arlot and Celisse (2010). The most popular method is K-fold cross-validation, which we introduce in Section 4.1. In Section 4.2, we discuss methods for cross-validation in the time-series setting.

4.1 K-fold cross-validation

For K-fold cross-validation, proposed by Geisser (1975), the data is split into K groups, referred to as folds, of approximately equal size. Let $\mathcal{K}_k$ denote the set of observations in the kth fold, and let n_k be the size of data partition k for $k = 1, ..., K$. In the kth step, the kth fold is treated as the validation data set and the remaining $K - 1$ folds constitute the training data set. The model is fit to the training data for a given value of λ and α. The resulting estimate, which is based on all the data except the observations in fold k, is $\hat{\boldsymbol{\beta}}_k(\lambda, \alpha)$. The procedure is repeated for each fold, as illustrated in Figure 2, so that every data point is used for validation once. The mean squared prediction error for each fold is computed as

$$\text{MSPE}_k(\lambda, \alpha) = \frac{1}{n_k} \sum_{i \in \mathcal{K}_k} \left(y_i - \boldsymbol{x}_i' \hat{\boldsymbol{\beta}}_k(\lambda, \alpha) \right)^2 .$$

4. We follow Chen and Chen (2008, p. 768) and use $\xi = 1 - \log(n)/(2\log(p))$ as the default choice. An upper and lower threshold is applied to ensure that ξ lies in the [0,1] interval.

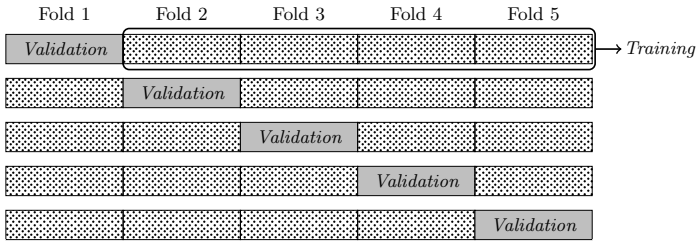


Figure 2: Data partition for 5-fold cross-validation. Each row corresponds to one step and each column to one data partition ('fold'). In the first step, fold 1 constitutes the validation data and folds 2-5 are the training data.

The K-fold cross-validation estimate of the MSPE, which serves as a measure of prediction performance, is

$$\hat{\mathcal{L}}^{CV}(\lambda, \alpha) = \frac{1}{K} \sum_{k=1}^{K} \text{MSPE}_k(\lambda, \alpha).$$

This suggests selecting λ and α as the values that minimize $\hat{\mathcal{L}}^{CV}(\lambda, \alpha)$. An alternative common rule is to use the largest value of λ that is within one standard deviation of the minimum, which leads to a more parsimonious model.

Cross-validation can be computationally expensive. It is necessary to compute $\hat{\mathcal{L}}^{CV}$ for each value of λ on a grid if α is fixed (e.g. when using the lasso) or, in the case of the elastic net, for each combination of values of λ and α on a two-dimensional grid. In addition, the model must be estimated K times at each grid point, such that the computational cost is approximately proportional to K.[5]

Standardization adds another layer of computational cost to K-fold cross validation. An important principle in cross-validation is that the training data set should not contain information from the validation dataset. This mimics the real-world situation where out-of-sample predictions are made not knowing what the true response is. The principle applies not only to individual observations, but also to data transformations such as mean-centering and standardization. Specifically, data transformations applied to the training data should not use information from the validation data or full dataset. Mean-centering and standardization using sample means and sample standard deviations for the full sample would violate this principle. Instead, when in each step the model is fit to the training data for a given λ and α, the training dataset must be re-centered and re-standardized, or, if standardization is built into the penalty loadings, the $\hat{\psi}_j$ must be recalculated based on the training dataset.

The choice of K is not only a practical problem; it also has theoretical implications. The variance of $\hat{\mathcal{L}}^{CV}$ decreases with K, and is minimal (for linear regression) if $K = n$,

5. An exception is the special case of leave-one-out cross-validation, where $K = n$. The advantage of LOO cross-validation for linear models is that there is a closed-form expression for the MSPE, meaning that the model needs to be estimated only once instead of n times.

which is referred to as leave-one-out or LOO CV. Similarly, the bias decreases with the size of the training data set. Given computational contraints, K between 5 and 10 are often recommended, arguing that the performance of CV rarely improves for K larger than 10 (Hastie et al. 2009; Arlot and Celisse 2010).

If the aim of the researcher's analysis is model identification rather than prediction, the theory requires training data to be 'small' and the evaluation sample to be close to n (Shao 1993, 1997). The reason is that more data is required to evaluate which model is the 'correct' one rather than to decrease bias and variance. This is referred to as *cross-validation paradox* (Yang 2006). However, since K-fold cross-validation sets the size of the training sample to approximately n/K, K-fold CV is necessarily ill-suited for selecting the true model.

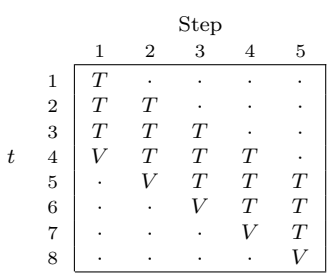
With regard to the K-fold cross-validated lasso, Chetverikov et al. (2019) demonstrate that the estimation error converges at a near-optimal rate to zero. For example, if the errors are Gaussian, the convergence rate is slower by a 'small logarithmic factor' of $\sqrt{\log(np)}$ relative to the theory-driven approach of regularization (to be discussed in Section 5). The authors also confirm through simulations, and by deriving sparsity bounds, that choosing the tuning parameter by cross-validation tends to yield more predictors than are in the true model and than when theory-driven regularization is employed.

4.2 Cross-validation with time-series data

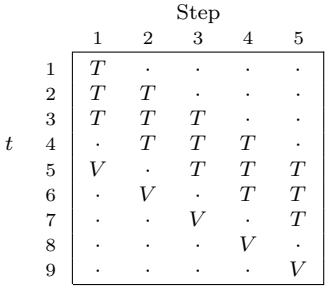
Serially dependent data violate the principle that training and validation data are independent. That said, standard K-fold cross-validation may still be appropriate in certain circumstances. Bergmeir et al. (2018) show that K-fold cross-validation remains valid in the pure auto-regressive model if one is willing to assume that the errors are uncorrelated. A useful implication is that K-fold cross-validation can be used on auto-regressive models that include a sufficient number of lags and that are not otherwise badly misspecified, since such models have uncorrelated errors.

Rolling h-step ahead CV is an intuitively appealing approach that directly incorporates the ordered nature of time series-data (Hyndman, Rob and Athanasopoulos 2018).[6] The procedure builds on repeated h-step ahead forecasts. The procedure is implemented in **lassopack** and illustrated in Figure 3-4.

6. Another approach is a variation of LOO cross-validation known as *h-block cross-validation* (Burman et al. 1994), which omits h observations between training and validation data.

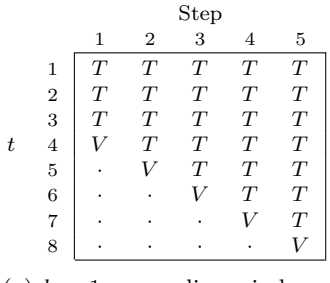


(a) $h = 1$, fixed window

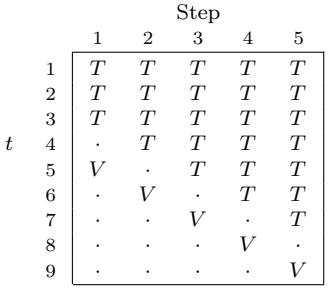


(b) $h = 2$, fixed window

Figure 4: Rolling h-step ahead cross-validation with fixed training window.



(a) $h = 1$, expanding window



(b) $h = 2$, expanding window

Figure 3: Rolling h-step ahead cross-validation with expanding training window. 'T' and 'V' denote that the observation is included in the training and validation sample, respectively. A dot ('.') indicates that an observation is excluded from both training and validation data.

Figure 3(a) corresponds to the default case of 1-step ahead cross-validation. 'T' denotes the observation included in the training sample and 'V' refers to the validation sample. In the first step, observations 1 to 3 constitute the training data set and observation 4 is the validation point, whereas the remaining observations are unused as indicated by a dot ('.'). Figure 3(b) illustrates the case of 2-step ahead cross-validation. In both cases, the training window expands incrementally, whereas Table 4 displays rolling CV with a fixed estimation window.

4.3 Comparison with information criteria

Since information-based approaches and cross-validation share the aim of model selection, one might expect that the two methods share some theoretical properties. Indeed, AIC and LOO-CV are asymptotically equivalent, as shown by Stone (1977) for fixed p. Since information criteria only require the model to be estimated once, they are

computationally much more attractive, which might suggest that information criteria are superior in practice. However, an advantage of CV is its flexibility and that it adapts better to situations where the assumptions underlying information criteria, e.g. homoskedasticity, are not satisfied (Andrews 1991; Arlot and Celisse 2010). If the aim of the analysis is identifying the true model, BIC and EBIC provide a better choice than K-fold cross-validation, as there are strong but well-understood conditions under which BIC and EBIC are model selection consistent.

5 Rigorous penalization

This section introduces the 'rigorous' or plug-in approach to penalization. Following Chernozhukov et al. (2016), we use the term 'rigorous' to emphasize that the framework is grounded in theory in the sense of providing penalization parameters that guarantee optimal rate of convergence for prediction and parameter estimation. The approach also yields models whose size is of the same order as the true model. Rigorous penalization is of special interest, as it provides the basis for methods to facilitate causal inference in the presence of many instruments and/or many control variables; these methods are the IV-Lasso (Belloni et al. 2012), the post-double-selection (PDS) estimator (Belloni et al. 2014a) and the post-regularization estimator (CHS) (Chernozhukov et al. 2015); all of which are implemented in our sister package **pdslasso** (Ahrens et al. 2018).

We discuss the conditions required to derive theoretical results for the lasso in Section 5.1. Sections 5.2-5.5 present feasible algorithms for rigorous penalization choices for the lasso and square-root lasso under i.i.d., heteroskedastic and cluster-dependent errors. Section 5.6 presents a related test for joint significance testing.

5.1 Theory of the lasso

There are three main conditions required to guarantee that the lasso is consistent in terms of prediction and parameter estimation.[7] The first condition relates to sparsity. Sparsity is an attractive assumption in settings where we have a large set of potentially relevant regressors, or consider various different model specifications, but assume that only one true model exists which includes a small number of regressors. We have introduced *exact sparsity* in Section 2, but the assumption is stronger than needed. For example, some true coefficients may be non-zero, but small in absolute size, in which case it might be preferable to omit them. For this reason, we use a weaker assumption:

Approximate sparsity. Belloni et al. (2012) consider the *approximate sparse model (ASM)*,

$$y_i = f(\boldsymbol{w}_i) + \varepsilon_i = \boldsymbol{x}_i'\boldsymbol{\beta}_0 + r_i + \varepsilon_i. \tag{6}$$

7. For a more detailed treatment, we recommend Hastie et al. (2015, Ch. 11) and Bühlmann and Van de Geer (2011).

The elementary regressors $\boldsymbol{w}_i$ are linked to the dependent variable through the unknown and possibly non-linear function $f(\cdot)$. The aim of the lasso (and square-root lasso) estimation is to approximate $f(\boldsymbol{w}_i)$ using the target parameter vector $\boldsymbol{\beta}_0$ and the transformations $\boldsymbol{x}_i := P(\boldsymbol{w}_i)$, where $P(\cdot)$ denotes a dictionary of transformations. The vector $\boldsymbol{x}_i$ may be large relative to the sample size, either because $\boldsymbol{w}_i$ itself is high-dimensional and $\boldsymbol{x}_i := \boldsymbol{w}_i$, or because a large number of transformations such as dummies, polynomials, interactions are considered to approximate $f(\boldsymbol{w}_i)$.

The assumption of approximate sparsity requires the existence of a target vector $\boldsymbol{\beta}_0$ which ensures that $f(\boldsymbol{w}_i)$ can be approximated sufficiently well, while using only a small number of non-zero coefficients. Specifically, the target vector $\boldsymbol{\beta}_0$ and the sparsity index s are assumed to meet the condition

$$\|\boldsymbol{\beta}_0\|_0 := s \ll n \qquad \text{with} \quad \frac{s^2 \log^2(p \vee n)}{n} \to 0,$$

and the resulting approximation error $r_i = f(\boldsymbol{w}_i) - \boldsymbol{x}_i'\boldsymbol{\beta}_0$ is bounded such that

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n} r_i^2} \leq C\sqrt{\frac{s}{n}}, \tag{7}$$

where C is a positive constant. To emphasize the distinction between approximate and exact sparsity, consider the special case where $f(\boldsymbol{w}_i)$ is linear with $f(\boldsymbol{w}_i) = \boldsymbol{x}_i'\boldsymbol{\beta}^\star$, but where the true parameter vector $\boldsymbol{\beta}^\star$ violates exact sparsity so that $\|\boldsymbol{\beta}^\star\|_0 > n$. If $\boldsymbol{\beta}^\star$ has many elements that are negligible in absolute size, we might still be able to approximate $\boldsymbol{\beta}^\star$ using the sparse target vector $\boldsymbol{\beta}_0$ as long as $r_i = \boldsymbol{x}_i'(\boldsymbol{\beta}^\star - \boldsymbol{\beta}_0)$ is sufficiently small as specified in (7).

Restricted sparse eigenvalue condition. The second condition relates to the Gram matrix, $n^{-1}\boldsymbol{X}'\boldsymbol{X}$. In the high-dimensional setting where p is larger than n, the Gram matrix is necessarily rank-deficient and the minimum (unrestricted) eigenvalue is zero, i.e.,

$$\min_{\boldsymbol{\delta}\neq 0} \frac{\|\boldsymbol{X}\boldsymbol{\delta}\|_2}{\sqrt{n}\,\|\boldsymbol{\delta}\|_2} = 0.$$

Thus, to accommodate large p, the full rank condition of OLS needs to be replaced by a weaker condition. While the full rank condition cannot hold for the full Gram matrix if $p > n$, we can plausibly assume that sub-matrices of size m are well-behaved. This is in fact the *restricted sparse eigenvalue (RSEC)* condition of Belloni et al. (2012). The RSEC formally states that the minimum sparse eigenvalues

$$\phi_{\min}(m) = \min_{1\leq\|\boldsymbol{\delta}\|_0\leq m} \frac{\boldsymbol{\delta}'\boldsymbol{X}'\boldsymbol{X}\boldsymbol{\delta}}{\|\boldsymbol{\delta}\|_2^2} \qquad \text{and} \qquad \phi_{\max}(m) = \max_{1\leq\|\boldsymbol{\delta}\|_0\leq m} \frac{\boldsymbol{\delta}'\boldsymbol{X}'\boldsymbol{X}\boldsymbol{\delta}}{\|\boldsymbol{\delta}\|_2^2}$$

are bounded away from zero and from above. The requirement $\phi_{\min}(m) > 0$ implies that all sub-matrices of size m have to be positive definite.[8]

8. The RSEC is stronger than required for the lasso. For example, Bickel et al. (2009) introduce the *restricted eigenvalue condition (REC)*. However, here we only present the RSEC which implies the

Regularization event. The third central condition concerns the choice of the penalty level λ and the predictor-specific penalty loadings ψ_j. The idea is to select the penalty parameters to control the random part of the problem in the sense that

$$\frac{\lambda}{n} \geq c \max_{1 \leq j \leq p} \left| \psi_j^{-1} S_j \right| \quad \text{where} \quad S_j = \frac{2}{n} \sum_{i=1}^{n} x_{ij} \varepsilon_i \tag{8}$$

with high probability. Here, $c > 1$ is a constant slack parameter and S_j is the jth element of the score vector $\boldsymbol{S} = \nabla \hat{Q}(\boldsymbol{\beta})$, the gradient of $\hat{Q}$ at the true value $\boldsymbol{\beta}$. The score vector summarizes the noise associated with the estimation problem.

Denote by $\Lambda = n \max_j |\psi_j^{-1} S_j|$ the maximal element of the score vector scaled by n and ψ_j, and denote by $q_\Lambda(\cdot)$ the quantile function for Λ.[9] In the rigorous lasso, we choose the penalty parameters λ and ψ_j and confidence level γ so that

$$\lambda \geq c q_\Lambda(1 - \gamma) \tag{9}$$

A simple example illustrates the intuition behind this approach. Consider the case where the true model has $\beta_j = 0$ for $j = 1, \ldots, p$, i.e., none of the regressors appear in the true model. It can be shown that for the lasso to select no variables, the penalty parameters λ and ψ_j need to satisfy $\lambda \geq 2 \max_j |\sum_i \psi_j^{-1} x_{ij} y_i|$.[10] Because none of the regressors appear in the true model, $y_i = \varepsilon_i$. We can therefore rewrite the requirement for the lasso to correctly identify the model without regressors as $\lambda \geq 2 \max_j |\sum_i \psi_j^{-1} x_{ij} \varepsilon_i|$, which is the regularization event in (8). We want this regularization event to occur with high probability of at least $(1 - \gamma)$. If we choose values for λ and ψ_j such that $\lambda \geq q_\Lambda(1 - \gamma)$, then by the definition of a quantile function we will choose the correct model—no regressors—with probability of at least $(1 - \gamma)$. This is simply the rule in (9).

The chief practical problem in using the rigorous lasso is that the quantile function $q_\Lambda(\cdot)$ is unknown. There are two approaches to addressing this problem proposed in the literature, both of which are implemented in `rlasso`. The `rlasso` default is the 'asymptotic' or *X-independent* approach: theoretically grounded and feasible penalty level and loadings are used that guarantee that (8) holds asymptotically, as $n \to \infty$ and $\gamma \to 0$. The *X-independent* penalty level choice can be interpreted as an asymptotic upper bound on the quantile function $q_\Lambda(.)$. In the 'exact' or *X-dependent* approach, the quantile function $q_\Lambda(.)$ is directly estimated by simulating the distribution of $q_\Lambda(1 - \gamma | \boldsymbol{X})$, the $(1-\gamma)$-quantile of Λ conditional on the observed regressors $\boldsymbol{X}$. We first focus on the *X-independent approach*, and introduce the *X-dependent approach* in Section 5.5.

5.2 Rigorous lasso

Belloni et al. (2012) show—using moderate deviation theory of self-normalized sums

REC and is sufficient for both lasso and post-lasso. Different variants of the REC and RSEC have been proposed in the literature; for an overview see Bühlmann and Van de Geer (2011).

9. That is, the probability that Λ is at most a is $q_\Lambda(a)$.
10. See, for example, Hastie et al. (2015, Ch. 2).

from Jing et al. (2003)—that the regularization event in (8) holds asymptotically, i.e.,

$$\mathrm{P}\left(\max_{1\leq j\leq p} c\left|S_j\right| \leq \frac{\lambda\psi_j}{n}\right) \to 1 \quad \text{as} \quad n\to\infty,\ \gamma\to 0. \tag{10}$$

if the penalty levels and loadings are set to

$$\begin{aligned} &\text{homoskedasticity:} \quad \lambda = 2c\sigma\sqrt{n}\Phi^{-1}(1-\gamma/(2p)), \qquad \psi_j = \sqrt{\frac{1}{n}\textstyle\sum_i x_{ij}^2}, \\ &\text{heteroskedasticity:} \quad \lambda = 2c\sqrt{n}\Phi^{-1}(1-\gamma/(2p)), \qquad \psi_j = \sqrt{\frac{1}{n}\textstyle\sum_i x_{ij}^2\varepsilon_i^2}, \end{aligned} \tag{11}$$

under homoskedasticity and heteroskedasticity, respectively. c is the slack parameter from above and the significance level γ is required to converge towards 0. `rlasso` uses $c = 1.1$ and $\gamma = 0.1/\log(n)$ as defaults.[11,12]

Homoskedasticity. We first focus on the case of homoskedasticity. In the rigorous lasso approach, we standardize the score. But since $E(x_{ij}^2\varepsilon_i^2) = \sigma E(x_{ij}^2)$ under homoskedasticity, we can separate the problem into two parts: the regressor-specific penalty loadings $\psi_j = \sqrt{(1/n)\sum_i x_{ij}^2}$ standardize the regressors, and σ moves into the overall penalty level. In the special case where the regressors have already been standardized such that $(1/n)\sum_i x_{ij}^2 = 1$, the penalty loadings are $\psi_j = 1$. Hence, the purpose of the regressor-specific penalty loadings in the case of homoskedasticity is to accommodate regressors with unequal variance.

The only unobserved term is σ, which appears in the optimal penalty level λ. To estimate σ, we can use some initial set of residuals $\hat{\varepsilon}_{0,i}$ and calculate the initial estimate as $\hat{\sigma}_0 = \sqrt{(1/n)\sum_i \hat{\varepsilon}_{0,i}^2}$. A possible choice for the initial residuals is $\hat{\varepsilon}_{0,i} = y_i$ as in Belloni et al. (2012) and Belloni et al. (2014a). `rlasso` uses the OLS residuals $\hat{\varepsilon}_{0,i} = y_i - \boldsymbol{x}_i[\mathcal{D}]'\hat{\boldsymbol{\beta}}_{OLS}$ where $\mathcal{D}$ is the set of 5 regressors exhibiting the highest absolute correlation with y_i.[13] The procedure is summarized in Algorithm A:

❑ **Algorithm A: Estimation of penalty level under homoskedasticity.**

1. Set $k = 0$, and define the maximum number of iterations, K. Regress y_i against the subset of d predictors exhibiting the highest correlation coefficient with y_i and compute the initial residuals as $\hat{\varepsilon}_{0,i} = \hat{\varepsilon}_{k,i} = y_i - \boldsymbol{x}_i[\mathcal{D}]'\hat{\boldsymbol{\beta}}_{OLS}$. Calculate the homoskedastic penalty loadings in (11).

11. The parameters c and γ can be controlled using the options `c(`*real*`)` and `gamma(`*real*`)`. Note that we need to choose c greater than 1 for the regularization event to hold asymptotically, but not too high as the shrinkage bias is increasing in c.
12. An alternative *X-independent* choice is to set $\lambda = 2c\sigma\sqrt{2n\log(2p/\gamma)}$. Since $\sqrt{n}\Phi^{-1}(1-\gamma/(2p)) \leq \sqrt{2n\log(2p/\gamma)}$, this will lead to a more parsimonious model, but also to a larger bias. To use the alternative *X-independent*, specify the `lalt` option.
13. This is also the default setting in Chernozhukov et al. (2016). The number of regressors used for calculating the initial residuals can be controlled using the `corrnumber(`*integer*`)` option, where 5 is the default and 0 corresponds to $\hat{\varepsilon}_{0,i} = y_i$.

2. If $k \leq K$, compute the homoskedastic penalty level in (11) by replacing σ with $\hat{\sigma}_k = \sqrt{(1/n)\sum_i \hat{\varepsilon}_{k,i}^2}$, and obtain the rigorous lasso or post-lasso estimator $\hat{\boldsymbol{\beta}}_k$. Update the residuals $\hat{\varepsilon}_{k+1,i} = y_i - \boldsymbol{x}_i'\hat{\boldsymbol{\beta}}_k$. Set $k \leftarrow k+1$.
3. Repeat step 2 until $k > K$ or until convergence by updating the penalty level.

❑

The `rlasso` default is to perform one further iteration after the initial estimate (i.e., $K = 1$), which in our experience provides good performance. Both lasso and post-lasso can be used to update the residuals. `rlasso` uses post-lasso to update the residuals.[14]

Heteroskedasticity. The *X-independent* choice for the overall penalty level under heteroskedasticity is $\lambda = 2c\sqrt{n}\Phi^{-1}(1-\gamma/(2p))$. The only difference with the homoskedastic case is the absence of σ. The variance of ε is now captured via the penalty loadings, which are set to $\psi_j = \sqrt{\frac{1}{n}\sum_i x_{ij}^2\varepsilon_i^2}$. Hence, the penalty loadings here serve two purposes: accommodating both heteroskedasticity and regressors with uneven variance.

To help with the intuition, we consider the case where the predictors are already standardized. It is easy to see that, if the errors are homoskedastic with $\sigma = 1$, the penalty loadings are (asymptotically) just $\psi_j = 1$. If the data are heteroskedastic, however, the standardized penalty loading will not be 1. In most practical settings, the usual pattern will be that $\hat{\psi}_j > 1$ for some j. Intuitively, heteroskedasticity typically increases the likelihood that the term $\max_j |\sum_i x_{ij}\varepsilon_i|$ takes on extreme values, thus requiring a higher degree of penalization through the penalty loadings.[15]

The disturbances ε_i are unobserved, so we obtain an initial set of penalty loadings $\hat{\psi}_j$ from an initial set of residuals $\hat{\varepsilon}_{0,i}$ similar to the i.i.d. case above. We summarize the algorithm for estimating the penalty level and loadings as follows:

❑ **Algorithm B: Estimation of penalty loadings under heteroskedasticity.**

1. Set $k = 0$, and define the maximum number of iterations, K. Regress y_i against the subset of d predictors exhibiting the highest correlation coefficient with y_i and compute the initial residuals as $\hat{\varepsilon}_{0,i} = \hat{\varepsilon}_{k,i} = y_i - \boldsymbol{x}_i[\mathcal{D}]'\hat{\boldsymbol{\beta}}_{OLS}$. Calculate the heteroskedastic penalty level λ in (11).
2. If $k \leq K$, compute the heterokedastic penalty loadings using the formula given in in (11) by replacing ε_i with $\hat{\varepsilon}_{k,i}$, obtain the rigorous lasso or post-lasso estimator $\hat{\boldsymbol{\beta}}_k$. Update the residuals $\hat{\varepsilon}_{k+1,i} = y_i - \boldsymbol{x}_i'\hat{\boldsymbol{\beta}}_k$. Set $k \leftarrow k+1$.
3. Repeat step 2 until $k > K$ or until convergence by updating the penalty loadings.

❑

14. The `lassopsi` option can be specified, if rigorous lasso residuals are preferred.
15. To get insights into the nature of heteroskedasticity, `rlasso` also calculates and returns the standardized penalty loadings

$$\hat{\psi}_j^S = \hat{\phi}_j \left(\sqrt{\frac{1}{n}\sum_i x_{ij}^2} \sqrt{\frac{1}{n}\hat{\varepsilon}_i^2} \right)^{-1},$$

which are stored in `e(sPsi)`.

Theoretical property. Under the assumptions SEC, ASM and if penalty level λ and the penalty loadings are estimated by Algorithm A or B, the lasso and post-lasso obey:[16]

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\boldsymbol{x_i'}\hat{\boldsymbol{\beta}}-\boldsymbol{x_i'}\boldsymbol{\beta}\right)^2}=O\left(\sqrt{\frac{s\log(p\vee n)}{n}}\right), \tag{12}$$

$$\|\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}\|_1=O\left(\sqrt{\frac{s^2\log(p\vee n)}{n}}\right), \tag{13}$$

$$\|\hat{\beta}\|_0=O\left(s\right) \tag{14}$$

The first relation in (12) provides an asymptotic bound for the prediction error, and the second relation in (13) bounds the bias in estimating the target parameter $\boldsymbol{\beta}$. Belloni et al. (2012) refer to the above convergence rates as *near-oracle* rates. If the identity of the s variables in the model were known, the prediction error would converge at the oracle rate $\sqrt{s/n}$. Thus, the logarithmic term $\log(p\vee n)$ can be interpreted as the cost of not knowing the true model. The final relation is a sparsity bound which shows the size of the estimated model does not diverge relative to the true model.

For comparison, the rates of the rigorous lasso are faster than the convergence rates of the K-fold cross-validated lasso derived in Chetverikov et al. (2019). Furthermore, the sparsity bound in Chetverikov et al. (2019) does not rule out situations where $\|\hat{\beta}\|_0-s\to\infty$, implying that the model estimated using cross-validation may be much larger than the true model. This is because the penalty level chosen by K-fold cross-validation cannot be guaranteed to satisfy the regularization event in (8). Indeed, Chetverikov et al. (2019) confirm in simulations that the cross-validated penalty is often smaller than required by (8).

5.3 Rigorous square-root lasso

The theory of the square-root lasso is similar to the theory of the lasso (Belloni et al. 2011, 2014b). The jth element of the score vector is now defined as

$$S_j=\frac{\frac{1}{n}\sum_{i=1}^{n}x_{ij}\varepsilon_i}{\left\{\frac{1}{n}\sum_{i=1}^{n}\varepsilon_i^2\right\}^{1/2}}. \tag{15}$$

To see why the square-root lasso is of special interest, we define the standardized errors ν_i as $\nu_i=\varepsilon_i/\sigma$. The jth element of the score vector becomes

$$S_j=\frac{\frac{1}{n}\sum_{i=1}^{n}x_{ij}\sigma\nu_i}{\left\{\frac{1}{n}\sum_{i=1}^{n}\sigma^2\nu_i^2\right\}^{1/2}}=\frac{\frac{1}{n}\sum_{i=1}^{n}x_{ij}\nu_i}{\left\{\frac{1}{n}\sum_{i=1}^{n}\nu_i^2\right\}^{1/2}} \tag{16}$$

16. For the sake of brevity, we omit additional technical conditions relating to the moments of the error and the predictors. These conditions are required to make use of the moderate deviation theory of self-normalized sums (Jing et al. 2003), which is employed to relax the assumption of Gaussian errors. See condition RF in Belloni et al. (2012) and condition SM in Belloni et al. (2014a).

and is thus independent of σ. For the same reason, the optimal penalty level for the square-root lasso in the i.i.d. case,

$$\lambda = c\sqrt{n}\Phi^{-1}(1-\gamma/(2p)), \tag{17}$$

is independent of the noise level σ.

Homoskedasticity. The ideal penalty loadings under homoskedasticity for the square-root lasso are given by formula (iv) in Table 1, which provides an overview of penalty loading choices. The ideal penalty parameters are independent of the unobserved error, which is an appealing theoretical property and implies a practical advantage. Since both λ and ψ_j can be calculated from the data, the rigorous square-root lasso is a one-step estimator under homoskedasticity. Belloni et al. (2011) show that the square-root lasso performs similarly to the lasso with infeasible ideal penalty loadings.

Heteroskedasticity. In the case of heteroskedasticity, the optimal square-root lasso penalty level remains (17), but the penalty loadings, given by formula (v) in Table 1, depend on the unobserved error and need to be estimated. Note that the updated penalty loadings using the residuals $\hat{\varepsilon}_i$ employ thresholding: the penalty loadings are enforced to be greater than or equal to the loadings in the homoskedastic case. The `rlasso` default algorithm used to obtain the penalty loadings in the heteroskedastic case is analogous to Algorithm B.[17] While the ideal penalty loadings are not independent of the error term if the errors are heteroskedastic, the square-root lasso may still have an advantage over the lasso, since the ideal penalty loadings are pivotal with respect to the error term up to scale, as pointed out above.

5.4 Rigorous penalization with panel data

Belloni et al. (2016) extend the rigorous framework to the case of clustered data, where a limited form of dependence—within-group correlation—as well as heteroskedasticity are accommodated. They prove consistency of the rigorous lasso using this approach in the large n, fixed T and large n, large T settings. The authors present the approach in the context of a fixed-effects panel data model, $y_{it} = \boldsymbol{x}_{it}'\boldsymbol{\beta} + \mu_i + \varepsilon_{it}$, and apply the rigorous lasso after the within transformation to remove the fixed effects μ_i. The approach extends to any clustered-type setting and to balanced and unbalanced panels. For convenience we ignore the fixed effects and write the model as a balanced panel:

$$y_{it} = \boldsymbol{x}_{it}'\boldsymbol{\beta} + \varepsilon_{it} \qquad i = 1, \ldots, n,\ t = 1, \ldots, T \tag{18}$$

The intuition behind the Belloni et al. (2016) approach is similar to that behind the clustered standard errors reported by various Stata estimation commands: observations

17. The `rlasso` default for the square-root lasso uses a first-step set of initial residuals. The suggestion of Belloni et al. (2014b) to use initial penalty loadings for regressor j of $\hat{\psi}_{0,j} = \max_i |x_{ij}|$ is available using the `maxabsx` option.

	lasso	square-root lasso
homoskedasticity	(i) $\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_{ij}^2}$	(iv) $\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_{ij}^2}$
heteroskedasticity	(ii) $\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_{ij}^2 \varepsilon_i^2}$	(v) $\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_{ij}^2} \vee \sqrt{\frac{\sum_{i=1}^{n} x_{ij}^2 \varepsilon_i^2}{\sum_{i=1}^{n} \varepsilon_i^2}}$
cluster-dependence	(iii) $\sqrt{\frac{1}{nT}\sum_{i=1}^{n} u_{ij}^2}$	(vi) $\sqrt{\frac{1}{nT}\sum_{i=1}^{n}\sum_{t}^{T} x_{ijt}^2} \vee \sqrt{\frac{\sum_{i=1}^{n} u_{ij}^2}{\sum_{i=1}^{n}\sum_{t=1}^{T} \varepsilon_{it}^2}}$

Note: Formulas (iii) and (vi) use the notation $u_{ij} = \sum_t x_{itj}\varepsilon_{it}$.

Table 1: Ideal penalty loadings for the lasso and square-root lasso under homoskedasticity, heteroskedasticity and cluster-dependence.

within clusters are aggregated to create 'super-observations' which are assumed independent, and these super-observations are treated similarly to cross-sectional observations in the non-clustered case. Specifically, define for the ith cluster and jth regressor the super-observation $u_{ij} := \sum_t x_{ijt}\varepsilon_{it}$. Then the penalty loadings for the clustered case are

$$\psi_j = \sqrt{\frac{1}{nT}\sum_{i=1}^{n} u_{ij}^2},$$

which resembles the heteroskedastic case. The `rlasso` default for the overall penalty level is the same as in the heteroskedastic case, $\lambda = 2c\sqrt{n}\Phi^{-1}(1-\gamma/(2p))$, except that the default value for γ is $0.1/\log(n)$, i.e., we use the number of clusters n rather than the number of observations nT. `lassopack` also implements the rigorous square-root lasso for panel data, which uses the overall penalty in (17) and the penalty loadings in formula (vi), Table 1.

5.5 X-dependent lambda

There is an alternative, sharper choice for the overall penalty level, referred to as the *X-dependent* penalty. Recall that the asymptotic, *X-independent* choice in (11) can be interpreted as an asymptotic upper bound on the quantile function of Λ, which is the scaled maximum value of the score vector. Instead of using the asymptotic choice, we can estimate by simulation the distribution of Λ conditional on the observed $\boldsymbol{X}$, and use this simulated distribution to obtain the quantile $q_\Lambda(1-\gamma|\boldsymbol{X})$.

In the case of estimation by the lasso under homoskedasticity, we simulate the dis-

	lasso	square-root lasso
homoskedasticity	(i) $2\hat{\sigma} \max_{1\leq j\leq p} \left\lvert \sum_{i=1}^{n} \psi_j^{-1} x_{ij} g_i \right\rvert$	(iv) $\frac{1}{\hat{\sigma}_g} \max_{1\leq j\leq p} \left\lvert \sum_{i=1}^{n} \psi_j^{-1} x_{ij} g_i \right\rvert$
heteroskedasticity	(ii) $2 \max_{1\leq j\leq p} \left\lvert \sum_{i=1}^{n} \psi_j^{-1} x_{ij} \hat{\varepsilon}_i g_i \right\rvert$	(v) $\frac{1}{\hat{\sigma}_g} \max_{1\leq j\leq p} \left\lvert \sum_{i=1}^{n} \psi_j^{-1} x_{ij} \hat{\varepsilon}_i g_i \right\rvert$
cluster-dependence	(iii) $2 \max_{1\leq j\leq p} \left\lvert \sum_{i=1}^{n} \psi_j^{-1} \hat{u}_{ij} g_i \right\rvert$	(vi) $\frac{1}{\hat{\sigma}_g} \max_{1\leq j\leq p} \left\lvert \sum_{i=1}^{n} \psi_j^{-1} \hat{u}_{ij} g_i \right\rvert$

Note: g_i is an *i.i.d.* standard normal variate drawn independently of the data; $\hat{\sigma}_g = \frac{1}{n}\sum_i g_i^2$. Formulas (iii) and (vi) use the notation $\hat{u}_{ij} = \sum_t x_{itj}\hat{\varepsilon}_{it}$.

Table 2: Definition of W statistic for the simulation of the distribution of Λ for the lasso and square-root lasso under homoskedasticity, heteroskedasticity and cluster-dependence.

tribution of Λ using the statistic W, defined as

$$W = 2\hat{\sigma} \max_{1\leq j\leq p} \left\lvert \sum_{i=1}^{n} \psi_j^{-1} x_{ij} g_i \right\rvert,$$

where ψ_j is the penalty loading for the homoskedastic case, $\hat{\sigma}$ is an estimate of the error variance using some previously-obtained residuals, and g_i is an *i.i.d.* standard normal variate drawn independently of the data. The *X-dependent* penalty choice is sharper and adapts to correlation in the regressor matrix (Belloni and Chernozhukov 2011).

Under heteroskedasticity, the lasso *X-dependent* penalty is obtained by a multiplier bootstrap procedure. In this case the simulated statistic W is defined as in formula (ii) in Table 2. The cluster-robust *X-dependent* penalty is again obtained analogously to the heteroskedastic case by defining super-observations, and the statistic W is defined as in formula (iii) in Table 2. Note that the standard normal variate g_i varies across but not within clusters.

The *X-dependent* penalties for the square-root lasso are similarly obtained from quantiles of the simulated distribution of the square-root lasso Λ, and are given by formulas (iv), (v) and (vi) in Table 2 for the homoskedastic, heteroskedastic and clustered cases, respectively.[18]

18. Note that since g_i is standard normal, in practice the term $\frac{1}{\hat{\sigma}_g}$ that appears in the expressions for the square-root lasso W will be approximately 1.

5.6 Significance testing with the rigorous lasso

Inference using the lasso, especially in the high-dimensional setting, is a challenging and ongoing research area of research (see Footnote 2). A test that has been developed and is available in `rlasso` corresponds to the test for joint significance of regressors using F or χ^2 tests that is common in regression analysis. Specifically, Belloni et al. (2012) suggest using the Chernozhukov et al. (2013, Appendix M) sup-score test to test for the joint significance of the regressors, i.e.,

$$H_0 : \beta_1 = \ldots = \beta_p = 0.$$

As in the preceding sections, regressors are assumed to be mean-centered and in their original units.

If the null hypothesis is correct and the rest of the model is well-specified, including the assumption that the regressors are orthogonal to the disturbance ε_i, then $y_i = \varepsilon_i$ and hence $E(\boldsymbol{x}_i\varepsilon_i) = E(\boldsymbol{x}_i y_i) = 0$. The sup-score statistic is

$$S_{SS} = \sqrt{n} \max_{1\leq j\leq p} \left| \frac{1}{n} \sum_{i=1}^{n} \psi_j^{-1} x_{ij} y_i \right| \tag{19}$$

where $\psi_j = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_{ij}y_i)^2}$. Intuitively, the ψ_j in (19) plays the same role as the penalty loadings do in rigorous lasso estimation.

The p-value for the sup-score test is obtained by a multiplier bootstrap procedure simulating the distribution of S_{SS} by the statistic W, defined as

$$W = \sqrt{n} \max_{1\leq j\leq p} \left| \frac{1}{n} \sum_{i=1}^{n} \psi_j^{-1} x_{ij} y_i g_i \right|,$$

where g_i is an *i.i.d.* standard normal variate drawn independently of the data and ψ_j is defined as in (19).

The procedure above is valid under both homoskedasticity and heteroskedasticity, but requires independence. A cluster-robust version of the sup-score test that accommodates within-group dependence is

$$S_{SS} = \sqrt{nT} \max_{1\leq j\leq p} \left| \frac{1}{nT} \sum_{i=1}^{n} \psi_j^{-1} u_{ij} \right| \tag{20}$$

where $u_{ij} := \sum_{t=1}^{T} x_{ijt} y_{it}$ and $\psi_j = \frac{1}{nT}\sum_{i=1}^{n} u_{ij}^2$.

The p-value for the cluster version of S_{SS} comes from simulating

$$W = \sqrt{nT} \max_{1\leq j\leq p} \left| \frac{1}{nT} \sum_{i=1}^{n} \psi_j^{-1} u_{ij} g_i \right|$$

where again g_i is an *i.i.d.* standard normal variate drawn independently of the data. Note again that g_i is invariant within clusters.

`rlasso` also reports conservative critical values for S_{SS} using an asymptotically-justified upper bound: $CV = c\Phi^{-1}(1 - \gamma_{SS}/(2p))$. The default value of the slack parameter is $c = 1.1$ and the default test level is $\gamma_{SS} = 0.05$. These parameters can be varied using the `c(`*real*`)` and `ssgamma(`*real*`)` options, respectively. The simulation procedure to obtain p-values using the simulated statistic W can be computationally intensive, so users can request reporting of only the computationally inexpensive conservative critical values by setting the number of simulated values to zero with the `ssnumsim(`*int*`)` option.

6 The commands

The package **lassopack** consists of three commands: `lasso2`, `cvlasso` and `rlasso`. Each command offers an alternative method for selecting the tuning parameters λ and α. We discuss each command in turn and present its syntax. We focus on the main syntax and the most important options. Some technical options are omitted for the sake of brevity. For a full description of syntax and options, we refer to the help files.

6.1 lasso2: Base command and information criteria

The primary purpose of `lasso2` is to obtain the solution of (adaptive) lasso, elastic net and square-root lasso for a single value of λ or a list of penalty levels, i.e., for $\lambda_1, \ldots, \lambda_r, \ldots, \lambda_R$, where R is the number of penalty levels. The basic syntax of `lasso2` is as follows:

`lasso2` *depvar indepvars* [*if*] [*in*] [`,` *options*]

We omit the full list of options for the sake of brevity. Selected options are discussed below and we refer to the help file for a full list of options.

Estimation methods

`alpha(`*real*`)` controls the elastic net parameter, α, which controls the degree of ℓ_1-norm (lasso-type) to ℓ_2-norm (ridge-type) penalization. `alpha(1)` corresponds to the lasso (the default estimator), and `alpha(0)` to ridge regression. The *real* value must lie in the interval [0,1].

`sqrt` specifies the square-root lasso estimator. Since the square-root lasso does not employ any form of ℓ_2-penalization, the option is incompatible with `alpha(`*real*`)`.

`adaptive` specifies the adaptive lasso estimator. The penalty loading for predictor j is set to $|\hat{\beta}_{j,0}|^{-\theta}$ where $\hat{\beta}_{j,0}$ is the OLS estimator or univariate OLS estimator if $p > n$. θ is the adaptive exponent, and can be controlled using the `adatheta(`*real*`)` option.

`adaloadings(`*matrix*`)` is a matrix of alternative initial estimates, $\hat{\beta}_{j,0}$, used for calculating adaptive loadings. For example, this could be the vector `e(b)` from an initial

`lasso2` estimation. The absolute value of $\hat{\beta}_{j,0}$ is raised to the power $-\theta$ (note the minus).

`adatheta(`*real*`)` is the exponent for calculating adaptive penalty loadings. The default is `adatheta(1)`

`ols` specifies that post-estimation OLS estimates are displayed and returned in `e(betas)` or `e(b)`.

The options `alpha(`*real*`)`, `sqrt`, `adaptive` and `ols` can be used to select elastic net, square-root lasso, adaptive lasso and post-estimation OLS, respectively. The default estimator of `lasso2` is the lasso, which corresponds to `alpha(1)`. The special case of `alpha(0)` yields ridge regression.

Options relating to lambda

`lambda(`*numlist*`)` controls the penalty level(s), λ_r, used for estimation. *numlist* is a scalar value or list of values in descending order. Each λ_r must be greater than zero. If not specified, the default list is used which, using Mata syntax, is defined by

`exp(rangen(log(lmax),log(lminratio*lmax),lcount))`,

where `lcount`, `lminratio` and `lmax` are defined below, `exp()` is the exponential function, `log()` is the natural logarithm, and `rangen(`a`,`b`,`n`)` creates a column vector going from a to b in n-1 steps (see the `mf_range` help file). Thus, the default list ranges from `lmax` to `lminratio*lmax` and `lcount` is the number of values. The distance between each λ_r in the sequence is the same on the logarithmic scale.

`lcount(`*integer*`)` is the number of penalty values, R, for which the solution is obtained. The default is `lcount(100)`.

`lmax(`*real*`)` is the maximum value penalty level, λ_1. By default, λ_1 is chosen as the smallest penalty level for which the model is empty. Suppose the regressors are mean-centered and standardized, then λ_1 is defined as $\max_j \frac{2}{n\alpha}\sum_{i=1}^{n}|x_{ij}y_i|$ for the elastic net and $\max_j \frac{1}{n\alpha}\sum_{i=1}^{n}|x_{ij}y_i|$ for the square-root lasso (see Friedman et al. 2010, Section 2.5).

`lminratio(`*real*`)` is the ratio of the minimum penalty level, λ_R, to maximum penalty level, λ_1. *real* must be between 0 and 1. Default is `lminratio(0.001)`.

The behaviour of `lasso2` depends on whether the *numlist* in `lambda(`*numlist*`)` is a scalar or of length greater than one. If *numlist* is a list of more than one value, the solution consists of a matrix of coefficient estimates which are stored in `e(betas)`. Each row in `e(betas)` corresponds to a distinct value of λ_r and each column to one of the predictors in *indepvars*. The 'path' of coefficient estimates over λ_r can be plotted using `plotpath(`*method*`)`, where *method* controls whether the coefficient estimates are plotted against lambda ('lambda'), the natural logarithm of lambda ('lnlambda') or the ℓ_1-norm ('norm'). If the *numlist* in `lambda(`*numlist*`)` is a scalar value, the solution is a vector of coefficient estimates which is stored in `e(b)`. The default behaviour of `lasso2` is to use

a list of 100 values.

Information criteria

`lic(`*string*`)` specifies that, after the first `lasso2` estimation using a list of penalty levels, the model that corresponds to the minimum information criterion will be estimated and displayer. 'aic', 'bic', 'aicc', and 'ebic' (the default) are allowed. However, the results are not stored in `e()`.

`postresults` is used in combination with `lic(`*string*`)`. `postresults` stores estimation results of the model selected by information criterion in `e()`.[19]

`ic(`*string*`)` controls which information criterion is shown in the output of `lasso2` when `lambda()` is a list. 'aic', 'bic', 'aicc', and 'ebic' (the default are allowed).

`ebicxi`(*real*) controls the ξ parameter of the EBIC. ξ needs to lie in the [0,1] interval. $\xi = 0$ is equivalent to the BIC. The default choice is $\xi = 1 - \log(n)/(2\log(p))$.

In addition to obtaining the coefficient path, `lasso2` calculates four information criteria (AIC, AIC_c, BIC and EBIC). These information criteria can be used for model selection by choosing the value of λ_r that yields the lowest value for one of the four information criteria. The `ic(`*string*`)` option controls which information criterion is shown in the output, where *string* can be replaced with 'aic', 'aicc', 'bic or 'ebic'. `lic(`*string*`)` displays the estimation results corresponding to the model selected by an information criterion. It is important to note that `lic(`*string*`)` will not store the results of the estimation. This has the advantage that the user can compare the results for different information criteria without the need to re-estimate the full model. To save the estimation results, the `postresults` option should be specified in combination with `lic(`*string*`)`.

Penalty loadings and standardisation

`notpen(`*varlist*`)` sets penalty loadings to zero for predictors in *varlist*. Unpenalized predictors are always included in the model.

`partial(`*varlist*`)` specified that variables in *varlist* are partialled out prior to estimation.

`ploadings(`*matrix*`)` is a row-vector of penalty loadings, and overrides the default standardization loadings. The size of the vector should equal the number of predictors (excluding partialled-out variables and excluding the constant).

`unitloadings` specifies that penalty loadings be set to a vector of ones; overrides the default standardization loadings.

`prestd` specifies that dependent variable and predictors are standardized prior to estimation rather than standardized "on the fly" using penalty loadings. See Section 9.2 for more details. By default the coefficient estimates are un-standardized (i.e., re-

19. This option was called `postest` in earlier versions of **lassopack**.

turned in original units).

`stdcoef` returns coefficients in standard deviation units, i.e., do not un-standardize. Only supported with `prestd` option.

Fixed effects and intercept

`fe` within-transformation is applied prior to estimation. The option requires the data in memory to be `xtset`.

`noconstant` suppress constant from estimation. Default behaviour is to partial the constant out (i.e., to center the regressors).

Plotting

`plotpath(`*string*`)` plots the coefficients path as a function of the ℓ_1-norm ('norm'), lambda ('lambda]) or the log of lambda ('lnlambda').

`plotvar(`*varlist*`)` list of variables to be included in the plot.

`plotopt(`*string*`)` additional plotting options passed on to line. For example, to turn the legend off, use `plotopt(legend(off))`.

`plotlabel` displays variable labels in graph.

Replay syntax

The replay syntax of `lasso2` allows for plotting and changing display options, without the need to re-run the full model. It can also be used to estimate the model using the value of λ selected by an information criterion. The syntax is given by:

`lasso2` [`,` `plotpath(`*string*`)` `plotvar(`*varlist*`)` `plotopt(`*string*`)` `plotlabel` `postresults` `lic(`*method*`)` `ic(`*method*`)`]

Prediction syntax

`predict` [*type*] *newvar* [*if*] [*in*] [`, xb residuals ols lambda(`*real*`)` `lid(`*integer*`)` `approx noisily postresults`]

`xb` computes predicted values (the default).

`residuals` computes residuals.

`ols` specifies that post-estimation OLS will be used for prediction.

If the previous `lasso2` estimation uses more than one penalty level (i.e. $R > 1$), the following options are applicable:

`lambda(`*real*`)` specifies that lambda value used for prediction.

`lid(`*integer*`)` specifies the index of the lambda value used for prediction.

`approx` specifies that linear approximation is used instead of re-estimation. Faster, but only exact if coefficient path is piece-wise linear.

`noisily` prompts display of estimation output if re-estimation required.

`postresults` stores estimation results in `e()` if re-estimation is used.

6.2 Cross-validation with cvlasso

`cvlasso` implements K-fold and h-step ahead rolling cross-validation. The syntax of `cvlasso` is:

`cvlasso` *depvar indepvars* [*if*] [*in*] [`,` *options*]

We only discuss selected options and refer to the help file for a full list of options.

Options for K-fold cross-validation

`alpha(`*numlist*`)` is a scalar elastic net parameter or an ascending list of elastic net parameters. Note that the `alpha()` option of `cvlasso` option accepts a *numlist*, while `lasso2` only accepts a scalar. If *numlist* is a list longer than 1, `cvlasso` cross-validates over λ_r with $r = 1, \ldots, R$ and α_m with $m = 1, \ldots, M$.

`nfolds(`*integer*`)` is the number of folds used for K-fold cross-validation. The default is `nfolds(10)`, or $K = 10$.

`foldvar(`*varname*`)` can be used to specify what fold (data partition) each observation lies in. *varname* is an integer variable with values ranging from 1 to K. If not specified, the fold variable is randomly generated such that each fold is of approximately equal size.

`savefoldvar(`*varname*`)` saves the fold variable variable in *varname*.

`seed(`*integer*`)` sets the seed for the generation of a random fold variable.

Options for h-step ahead rolling cross-validation

`rolling` uses rolling h-step ahead cross-validation. The option requires the data to be `tsset` or `xtset`.

`h(`*integer*`)` changes the forecasting horizon. The default is `h(1)`.

`origin(`*integer*`)` controls the number of observations in the first training dataset.

`fixedwindow` ensures that the size of the training data set is constant.

Options for selection of lambda

`lopt` specifies that, after cross-validation, `lasso2` estimates the model with the value of λ_r that minimizes the mean-squared prediction error. That is, the model is estimated with $\lambda = \hat{\lambda}_{\texttt{lopt}}$.

`lse` specifies that, after cross-validation, `lasso2` estimates model with largest λ_r that is within one standard deviation from $\hat{\lambda}_{\texttt{lopt}}$. That is, the model is estimated with $\lambda = \hat{\lambda}_{\texttt{lse}}$.

`postresults` stores the `lasso2` estimation results in `e()` (to be used in combination with `lse` or `lopt`).

Plotting

`plotcv` plots the estimated mean-squared prediction error as a function of ln(lambda)

`plotopt(`*string*`)` additional plotting options passed on to line. For example, to turn the legend off, use `plotopt(legend(off))`.

Storing estimation results for each fold

`saveest(`*string*`)` saves `lasso2` results from each step of the cross-validation in *string*1, ..., *string*K where K is the number of folds. Intermediate results can be restored using `estimates restore`.

Replay syntax

The replay syntax of `cvlasso` is given by:

`cvlasso` [`, lopt lse plotcv(`*method*`) plotopt(`*string*`) postresults`]

Similar to `lasso2`, `cvlasso` also provides a replay syntax, which helps to avoid time-consuming re-estimations. The replay syntax of `cvlasso` can be used for plotting and to estimate the model corresponding to $\hat{\lambda}_{\texttt{lopt}}$ or $\hat{\lambda}_{\texttt{lse}}$.

Predict syntax

`predict` [*type*] *newvar* [*if*] [*in*] [`, xb residuals lopt lse noisily`]

6.3 rlasso: Rigorous penalization

`rlasso` implements theory-driven penalization for lasso and square-root lasso. It allows for heteroskedastic, cluster-dependent and non-Gaussian errors. Unlike `lasso2` and `cvlasso`, `rlasso` estimates the penalty level λ using iterative algorithms. The syntax

of `rlasso` is given by:

`rlasso` *depvar indepvars* [*if*] [*in*] [*weight*] [`,` *options*]

As above, we only discuss selected options and refer to the help file for a full list of options.

`pnotpen(`*varlist*`)` specifies that variables in *varlist* are not penalized.[20]

`robust` specifies that the penalty loadings account for heteroskedasticity.

`cluster(`*varname*`)` specifies that the penalty loadings account for clustering on variable *varname*.

`center` center moments in heteroskedastic and cluster-robust loadings.[21]

`lassopsi` use lasso or square-root lasso residuals to obtain penalty loadings. The default is post-estimation OLS.[22]

`corrnumber(`*integer*`)` number of high-correlation regressors used to obtain initial residuals. The default is `corrnumber(5)`, and `corrnumber(0)` implies that *depvar* is used in place of residuals.

`prestd` standardize data prior to estimation. The default is standardization during estimation via penalty loadings.

Options relating to lambda

`xdependent` specifies that the *X-dependent* penalty level is used; see Section 5.5.

`numsim(`*integer*`)` is the number of simulations used for the *X-dependent* case. The default is 5,000.

`lalternative` specifies the alternative, less sharp penalty level, which is defined as $2c\sqrt{2n\log(2p/\gamma)}$ (for the square-root lasso, $2c$ is replaced with c). See Footnote 12.

`gamma(`*real*`)` is the 'γ' in the rigorous penalty level (default $\gamma = 1/\log(n)$; cluster-lasso default $\gamma = 1/\log(n_{clust})$). See Equation (11).

`c(`*real*`)` is the 'c' in the rigorous penalty level. The default is `c(1.1)`. See Equation (11).

Sup-score test

`supscore` reports the sup-score test of statistical significance.

20. This option differs from that of `notpen(`*varlist*`)` as used with `cvlasso` and `lasso2`; see the discussion in Section 9.
21. For example, the uncentered heteroskedastic loading for regressor j is $\hat{\psi}_j = \sqrt{\frac{1}{n}\sum_i x_{ij}^2 \hat{\varepsilon}_i^2}$. In theory, $x_{ij}\varepsilon_i$ should be mean-zero. The centered penalty loading is $\hat{\psi}_j = \sqrt{\frac{1}{n}\sum_i (x_{ij}\hat{\varepsilon}_i - \hat{\mu})^2}$ where $\hat{\mu} = \frac{1}{n}\sum_i x_{ij}\hat{\varepsilon}_i$.
22. The option was called `lassoups` in earlier versions.

`testonly` reports only the sup-score test without lasso estimation.

`ssgamma(`*real*`)` is the test level for the conservative critical value for the sup-score test (default = 0.05, i.e., 5% significance level).

`ssnumsim(`*integer*`)` controls the number of simulations for sup-score test multiplier bootstrap. The default is 500, while 0 implies no simulation.

Predict syntax

`predict` [*type*] *newvar* [*if*] [*in*] [`, xb residuals lasso ols`]

`xb` generate fitted values (default).

`residuals` generate residuals.

`lasso` use lasso coefficients for prediction (default is to use estimates posted in `e(b)` matrix).

`ols` use OLS coefficients based on lasso-selected variables for prediction (default is to use estimates posted in `e(b)` matrix).

7 Demonstrations

In this section, we demonstrate the use of `lasso2`, `cvlasso` and `rlasso` using one cross-section example (in Section 7.1) and one time-series example (in Section 7.2).

7.1 Cross-section

For demonstration purposes, we consider the Boston Housing Dataset available on the UCI Machine Learning Repository.[23] The data set includes 506 observations and 14 variables.[24] The purpose of the analysis is to predict house prices using a set of census-level characteristics.

23. The dataset is available at https://archive.ics.uci.edu/ml/machine-learning-databases/housing/, or in CSV format via our website at https://statalasso.github.io/dta/housing.csv.
24. The following variables are included: per capita crime rate (`crim`), proportion of residential land zoned for lots over 25,000 sq.ft. (`zn`), proportion of non-retail business acres per town (`indus`), Charles River dummy variable (`chas`), nitric oxides concentration (parts per 10 million) (`nox`), average number of rooms per dwelling (`rm`), proportion of owner-occupied units built prior to 1940 (`age`), weighted distances to five Boston employment centres (`dis`), index of accessibility to radial highways (`rad`), full-value property-tax rate per $10,000 (`tax`), pupil-teacher ratio by town (`pratio`), $1000(Bk-0.63)^2$ where Bk is the proportion of blacks by town (`b`), % lower status of the population (`lstat`), median value (`medv`).

Estimation with lasso2

After importing the data and changing the variable order (to simplify the syntax), we employ the lasso estimator by regressing the outcome variable (`medv`) against 11 continuous predictors (`crim-lstat`), the binary Charles River dummy `chas` and the highway accessibility index `rad`, which we treat as categorical.

```
. insheet using https://statalasso.github.io/dta/housing.csv, comma
. order rad chas medv, last
. lasso2 medv crim-lstat i.chas ibn.rad
```

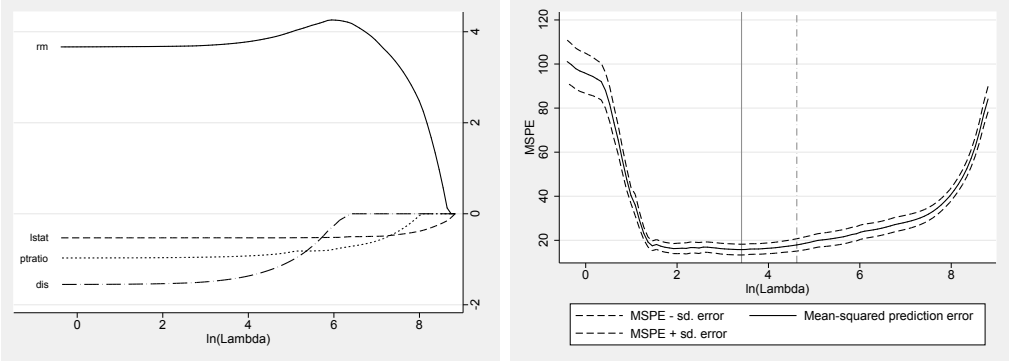
Knot	ID	Lambda	s	L1-Norm	EBIC	R-sq	Action
1	1	6858.98553	1	0.00000	2250.74087	0.0000	Added _cons.
2	2	6249.65216	2	0.08440	2207.91748	0.0924	Added lstat.
3	3	5694.45029	3	0.28098	2166.62026	0.1737	Added rm.
4	10	2969.09110	4	2.90443	1902.66627	0.5156	Added ptratio.
5	20	1171.07071	5	4.79923	1738.09475	0.6544	Added b.
6	22	972.24348	6	5.15524	1727.95402	0.6654	Added 1.chas.
7	26	670.12972	8	6.61915	1714.50618	0.6821	Added crim 3.rad.
8	28	556.35346	9	7.50948	1708.39482	0.6897	Added dis.
9	29	506.92856	10	8.07318	1706.78871	0.6945	Added 1bn.rad.
10	30	461.89442	11	8.77706	1705.92140	0.6987	Added nox.
11	32	383.47286	12	12.23038	1695.88184	0.7083	Added 8.rad.
12	33	349.40619	14	14.00603	1700.68964	0.7126	Added zn 6.rad.
13	37	240.83213	15	20.06993	1679.95704	0.7276	Added 7.rad.
14	38	219.43727	16	21.51820	1681.26608	0.7302	Added 4.rad.
15	41	165.99625	17	25.38355	1676.50748	0.7360	Added 24.rad.
16	44	125.57007	18	29.26833	1673.76687	0.7406	Added indus.
17	46	104.25048	20	31.35389	1681.69697	0.7429	Added tax 2.rad.
18	49	78.86167	19	34.29335	1669.60849	0.7459	Removed indus.
19	67	14.77724	20	41.36425	1668.26164	0.7497	Added indus.
20	88	2.09464	21	43.29218	1674.08431	0.7499	Added age.

```
Use long option for full output.
Type e.g. lasso2, lic(ebic) to run the model selected by EBIC.
```

The `lasso2` output above shows the following columns:

- `Knot` is the knot index. Knots are points at which predictors enter or leave the model. The default output shows one line per knot. If the `long` option is specified, one row per λ_r value is shown.
- `ID` shows the λ_r index, i.e., r. By default, `lasso2` uses a descending sequence of 100 penalty levels.
- `s` is the number of predictors in the model.
- `L1-Norm` shows the ℓ_1-norm of coefficient estimates.
- The sixth column (here labelled `EBIC`) shows one out of four information criteria. The `ic(`*string*`)` option controls which information criterion is displayed, where *string* can be replaced with 'aic', 'aicc', 'bic', and 'ebic' (the default).
- `R-sq` shows the R^2 value.
- The final column shows which predictors are entered or removed from the model at each knot. The order in which predictors are entered into the model can be interpreted as an indication of the relative predictive power of each predictor.

Since `lambda(`*real*`)` is not specified, `lasso2` obtains the coefficient path for a default



(a) Coefficient path (b) Mean squared prediction error

Figure 5: The left graph shows the coefficient path of the lasso for selected variables as a function of $\ln(\lambda)$. The right graph shows the mean squared prediction error estimated by cross-validation along with $\pm$ one standard error. The continuous and dashed vertical lines correspond to `lopt` and `lse`, respectively.

list of λ_r values. The largest penalty level is 6858.99, in which case the model does only include the constant.

We use the `ibn.`-prefix for `rad` instead of the `i.`-prefix commonly used in regression analysis. The `i.`-prefix triggers Stata to omit the (automatically selected) base category from the estimation to avoid perfect collinearity (see `help fvvarlist`). In contrast to OLS, the lasso does not require the full rank condition, allowing us to include one dummy for every category without removing the base category dummy. In this way we let the lasso choose the base category automatically as the categories omitted from the selected model.

Figure 5a shows the coefficient path of the lasso for selected variables as a function of $\ln(\lambda)$. The graph illustrates how the lasso shrinks coefficients towards zero as the penalty level increases (from left to right). We can also infer the order in which predictors leave the model. The figure was created using the following command:

```
. lasso2 medv crim-lstat i.chas ibn.rad, plotpath(lnlambda) ///
                        plotopt(legend(off) scheme(sj)) ///
                        plotlabel plotvar(rm lstat ptratio dis)
```

The `plotpath()` option passes settings on to Stata's `line` command, and is here used to switch the legend off. Instead of a standard legend, we specify `plotlabel` to show variable names on the left-hand side of the graph.

The commands of **lassopack** support Stata's factor variable notation. This is especially useful when the relationship between outcome variable and predictors is suspected to be non-linear. For demonstration purposes, we add squared terms and interaction effects. We also allow the effect of `lstat` to depend on whether the binary Charles River dummy (`chas`) takes the values one or zero.

```
. lasso2 medv c.(crim-b)##c.(crim-b) i.chas#(c.(lstat)##c.(lstat)) ibn.rad
```

`lasso2` supports model selection using information criteria. To this end, we use the replay syntax in combination with the `lic()` option, which avoids that the full model needs to be estimated again. The `lic()` option can also be specified in the first `lasso2` call. In the following example, the replay syntax works similar to a post-estimation command.

```
. lasso2, lic(ebic)
Use lambda=30.28269353185128 (selected by EBIC).
```

Selected	Lasso	Post-est OLS
dis	-0.8041913	-2.3591892
c.crim#c.crim	0.0022373	0.0041338
c.crim#c.nox	-0.3327808	-0.6752652
c.crim#c.rm	-0.0011237	0.0071626
c.crim#c.dis	-0.0076421	0.0113707
c.crim#c.b	-0.0003053	-0.0003214
c.zn#c.zn	0.0001680	0.0001673
c.zn#c.indus	-0.0053106	-0.0059749
c.indus#c.dis	-0.0642977	-0.0307386
c.indus#c.b	0.0007009	0.0004931
c.nox#c.rm	-3.0615123	-3.2526189
c.rm#c.rm	1.0155778	1.2150564
c.rm#c.age	-0.0004183	-0.0014745
c.rm#c.tax	-0.0018132	-0.0020079
c.rm#c.ptratio	-0.3798118	-0.5203655
c.rm#c.b	0.0004661	0.0003260
c.dis#c.dis	0.0326574	0.1500554
c.dis#c.tax	-0.0007936	-0.0006166
c.ptratio#		
c.ptratio	0.0432489	0.0692687
chas#c.lstat		
0	-1.0242696	-1.1353271
1	-0.6000422	-0.6461237
chas#c.lstat#		
c.lstat		
0	0.0151466	0.0185963
rad		
1	-2.9615120	-3.0500614
2	-2.5006144	-2.6742969
3	1.2399621	1.5646186
4	-0.1010378	-0.1689010
7	1.9427202	2.0772918
8	0.5591166	0.8479509
24	6.7270351	7.9864293
Partialled-out*		
_cons	37.1278484	42.1907216

Two columns are shown in the output; one for the lasso estimator and one for post-estimation OLS, which applies OLS to the model selected by the lasso.

K-fold cross-validation with cvlasso

Next, we consider K-fold cross-validation.

```
. set seed 123
. cvlasso medv c.(crim-b)##c.(crim-b) i.chas#(c.(lstat)##c.(lstat)) ibn.rad
K-fold cross-validation with 10 folds. Elastic net with alpha=1.
Fold 1 2 3 4 5 6 7 8 9 10
             |         Lambda           MSPE       st. dev.
-------------+------------------------------------------------
           1 |      6677.7298      84.121897      5.7688385
 (Output ommitted)
          46 |      101.49556      17.918966      2.7965745  ^
 (Output ommitted)
          59 |      30.282694      15.792392      2.4344696  *
 (Output ommitted)
         100 |      .66777298      101.10391      9.7304136
* lopt = the lambda that minimizes MSPE.
  Run model: cvlasso, lopt
^ lse = largest lambda for which MSPE is within one standard error of the minimal MSPE.
  Run model: cvlasso, lse
```

The `cvlasso` output displays four columns: the index of λ_r (i.e., r), the value of λ_r, the estimated mean squared prediction error, and the standard deviation of the mean squared prediction error. The output indicates the value of λ_r that corresponds to the lowest MSPE with an asterisk (`*`). We refer to this value as $\hat{\lambda}_{\texttt{lopt}}$. In addition, the symbol `^` marks the largest value of λ that is within one standard error of $\hat{\lambda}_{\texttt{lopt}}$, which we denote as $\hat{\lambda}_{\texttt{lse}}$.

The mean squared prediction is shown in Figure 5b, which was created using the `plotcv` option. The graph shows the mean squared prediction error estimated by cross-validation along with $\pm$ one standard error. The continuous and dashed vertical lines correspond to $\hat{\lambda}_{\texttt{lopt}}$ and $\hat{\lambda}_{\texttt{lse}}$, respectively.

To estimate the model corresponding to either $\hat{\lambda}_{\texttt{lopt}}$ or $\hat{\lambda}_{\texttt{lse}}$, we use the `lopt` or `lse` option, respectively. Similar to the `lic()` option of `lasso2`, `lopt` and `lse` can either specified in the first `cvlasso` call or after estimation using the replay syntax as in this example:

```
. cvlasso, lopt
Estimate lasso with lambda=30.283 (lopt).
```

Selected	Lasso	Post-est OLS
dis	-0.8041913	-2.3591892
c.crim#c.crim	0.0022373	0.0041338
c.crim#c.nox	-0.3327808	-0.6752652
c.crim#c.rm	-0.0011237	0.0071626
c.crim#c.dis	-0.0076421	0.0113707
c.crim#c.b	-0.0003053	-0.0003214
c.zn#c.zn	0.0001680	0.0001673
c.zn#c.indus	-0.0053106	-0.0059749
c.indus#c.dis	-0.0642977	-0.0307386
c.indus#c.b	0.0007009	0.0004931

```
        c.nox#c.rm |       -3.0615123     -3.2526189
         c.rm#c.rm |        1.0155778      1.2150564
        c.rm#c.age |       -0.0004183     -0.0014745
        c.rm#c.tax |       -0.0018132     -0.0020079
    c.rm#c.ptratio |       -0.3798118     -0.5203655
          c.rm#c.b |        0.0004661      0.0003260
      c.dis#c.dis |        0.0326574      0.1500554
      c.dis#c.tax |       -0.0007936     -0.0006166
        c.ptratio# |
        c.ptratio |        0.0432489      0.0692687
     chas#c.lstat |
               0 |       -1.0242696     -1.1353271
               1 |       -0.6000422     -0.6461237
     chas#c.lstat# |
          c.lstat |
               0 |        0.0151466      0.0185963
             rad |
               1 |       -2.9615120     -3.0500614
               2 |       -2.5006144     -2.6742969
               3 |        1.2399621      1.5646186
               4 |       -0.1010378     -0.1689010
               7 |        1.9427202      2.0772918
               8 |        0.5591166      0.8479509
              24 |        6.7270351      7.9864293
-----------------+---------------------------------
   Partialled-out* |
-----------------+---------------------------------
           _cons |       37.1278484     42.1907216
-------------------------------------------------
```

Rigorous penalization with rlasso

Lastly, we consider `rlasso`. The program `rlasso` runs an iterative algorithm to estimate the penalty level and loadings. In contrast to `lasso2` and `cvlasso`, it reports the selected model directly.

```
. rlasso medv c.(crim-b)##c.(crim-b) i.chas#(c.(lstat)##c.(lstat)) ibn.rad, supscore
```

```
-------------------------------------------------
        Selected |            Lasso   Post-est OLS
-----------------+-------------------------------
         ptratio |       -0.3730028     -0.3259609
     c.crim#c.rm |       -0.0036642     -0.0166472
   c.indus#c.dis |       -0.0222003     -0.0400010
 c.nox#c.ptratio |       -0.2563088     -0.5552952
       c.rm#c.rm |        0.3677334      0.4101983
        c.rm#c.b |        0.0011409      0.0013808
     c.age#c.dis |       -0.0004545     -0.0096741
     c.dis#c.tax |       -0.0001198     -0.0015891
    chas#c.lstat |
               0 |       -0.3295249     -0.3263231
           _cons |*      19.7449405     24.5139615
-------------------------------------------------
*Not penalized
Sup-score test H0: beta=0
CCK sup-score statistic  16.15 p-value= 0.000
CCK 5% critical value     3.75 (asympt bound)
```

The `supscore` option prompts the sup-score test of joint significance. The p-value is obtained through multiplier bootstrap. The test statistic of 16.15 can also be compared to the asymptotic 5% critical value (here 3.75).

7.2 Time-series data

A standard problem in time-series econometrics is to select an appropriate lag length. In this sub-section, we show how **lassopack** can be employed for this purpose. We consider Stata's built-in data set `lutkepohl2.dta`, which includes quarterly (log-differenced) consumption (`dln_consump`), investment (`dln_inv`) and income (`dln_inc`) series for West Germany over the period 1960, Quarter 1 to 1982, Quarter 4. We demonstrate both lag selection via information criteria and by h-step ahead rolling cross-validation. We do not consider the rigorous penalization approach of `rlasso` due to the assumption of independence, which seems too restrictive in the time-series context.

Information criteria

After importing the data, we run the most general model with up to 12 lags of `dln_consump`, `dln_inv` and `dln_inc` using `lasso2` with `lic(aicc)` option.

```
. webuse lutkepohl2, clear
(Quarterly SA West German macro data, Bil DM, from Lutkepohl 1993 Table E.1)
. lasso2 dln_consump L(1/12).(dln_inv dln_inc dln_consump), lic(aicc) long
```

Knot	ID	Lambda	s	L1-Norm	AICc	R-sq	Action
1	1	0.52531	1	0.00000	-714.43561	0.0000	Added _cons.
(Output ommitted)							
	11	0.20719	10	0.67593	-722.62355*	0.3078	
(Output ommitted)							
	100	0.00005	37	4.92856	-665.31816	0.6719	

```
*indicates minimum AICc.
Use lambda=.2071920751852477 (selected by AICC).

Use lambda=.2071920751852477 (selected by AICC).
```

Selected	Lasso	Post-est OLS
dln_inv		
L2.	0.0279780	0.0513004
dln_inc		
L1.	0.0672531	0.1522251
L2.	0.1184912	0.1675746
L3.	0.0779780	0.1261940
L8.	-0.1091959	-0.2481821
dln_consump		
L2.	0.0259311	0.0935048
L3.	0.0765755	0.1405377
L10.	0.0833425	0.2320500
L11.	-0.0891871	-0.1442602
Partialled-out*		
_cons	0.0133270	0.0079518

The output consists of two parts. The second part of the output is prompted since `lic(aicc)` is specified. `lic(aicc)` asks `lasso2` to estimate the model selected by AIC_c, which in this case corresponds to $\lambda_{11} = 0.207$.

h-step ahead rolling cross-validation

In the next step, we consider h-step ahead rolling cross-validation.

```
. cvlasso dln_consump L(1/12).(dln_inv dln_inc dln_consump), rolling
Rolling forecasting cross-validation with 1-step ahead forecasts. Elastic net with alpha=1.
Training from-to (validation point): 13-80 (81), 13-81 (82), 13-82 (83), 13-83 (84),
13-84 (85), 13-85 (86), 13-86 (87), 13-87 (88), 13-88 (89), 13-89 (> 90), 13-90 (91).
```

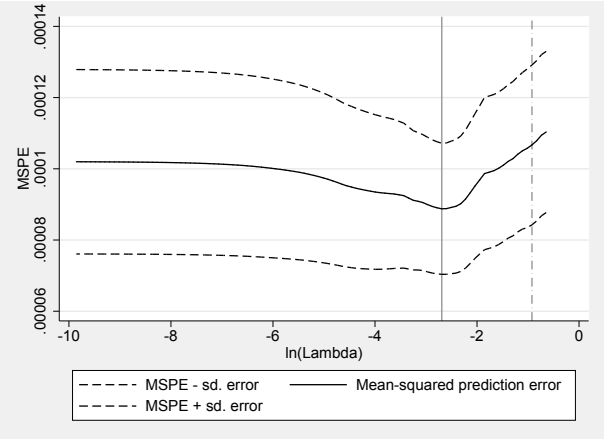


Figure 6: Cross-validation plot. The graph uses Stata's built-in data set `lutkepohl2.dta` and 1-step ahead rolling cross-validation with `origin(50)`.

The output indicates how the data set is partitioned into training data and the validation point. For example, the short-hand `13-80 (81)` in the output above indicates that observations 13 to 80 constitute the training data set in the first step of cross-validation, and observation 81 is the validation point. The options `fixedwindow`, `h(`*integer*`)` and `origin(`*integer*`)` can be used to control the partitioning of data into training and validation data. `h(`*integer*`)` sets the parameter h. For example, `h(2)` prompts 2-step ahead forecasts (the default is `h(1)`). `fixedwindow` ensures that the training data set is of same size in each step. If `origin(50)` is specified, the first training partition includes observations 13 to 50, as shown in the next example:

```
. cvlasso dln_consump L(1/12).(dln_inv dln_inc dln_consump), rolling origin(50) plotcv
Rolling forecasting cross-validation with 1-step ahead forecasts. Elastic net with alpha=1.
Training from-to (validation point): 13-50 (51), 13-51 (52), 13-52 (53), 13-53 (54),
  (Output omitted.)
13-82 (83), 13-83 (84), 13-84 (85), 13-85 (86), 13-86 (87), 13-87 (88), 13-88 (89),
13-89 (90), 13-90 (91).
```

The option `plotcv` creates the graph of the estimated mean squared prediction in Figure 6. To estimate the model corresponding to $\hat{\lambda}_{\texttt{lse}}$, we can as in the previous

examples use the replay syntax:

```
. cvlasso, lse
Estimate lasso with lambda=.397 (lse).
```

Selected	Lasso	Post-est OLS
dln_inv		
L2.	0.0071068	0.0481328
dln_inc		
L2.	0.0558422	0.2083321
L3.	0.0253076	0.1479925
dln_consump		
L3.	0.0260573	0.1079076
L11.	-0.0299307	-0.1957719
Partialled-out*		
_cons	0.0168736	0.0126810

We point out that care should be taken when setting the parameters of h-step ahead rolling cross-validation. The default settings have no particular econometric justification.

8 Monte Carlo Simulation

We have introduced three alternative approaches for setting the penalization parameters in Sections 3-5. In this section, we present results of Monte Carlo simulations which assess the performance of these approaches in terms of in-sample fit, out-of-sample prediction, model selection and sparsity. To this end, we generate artificial data using the process

$$y_i = 1 + \sum_{j=1}^{p} \beta_j x_{ij} + \varepsilon_i, \quad \varepsilon_i \sim \mathcal{N}(0, \sigma^2), \quad i = 1, \dots, 2n, \tag{21}$$

with $n = 200$. We report results for $p = 100$ and for the high-dimensional setting with $p = 220$. The predictors x_{ij} are drawn from a multivariate normal distribution with $\text{corr}(x_{ij}, x_{ir}) = 0.9^{|j-r|}$. We vary the noise level σ between 0.5 and 5; specifically, we consider $\sigma = \{0.5, 1, 2, 3, 5\}$. We define the parameters as $\beta_j = \mathbb{1}\{j \leq s\}$ for $j = 1, \dots, p$ with $s = 20$, implying exact sparsity. This simple design allows us to gain insights into the model selection performance in terms of false positive (the number of variables falsely included) and false negative frequency (the number of variables falsely omitted) when relevant and irrelevant regressors are correlated. All simulations use at least 1,000 iterations. We report additional Monte Carlo results in Appendix A.1, where we employ a design in which coefficients alternate in sign.

Since the aim is to assess in-sample and out-of-sample performance, we generate $2n$

observations, and use the data $i = 1, \ldots, n$ as the estimation sample and the observations $i = n+1, \ldots, 2n$ for assessing out-of-sample prediction performance. This allows us to calculate the root mean squared error (RMSE) and root mean squared prediction error (RMSPE) as

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_{i,n})^2} \qquad \text{and} \qquad \text{RMSPE} = \sqrt{\frac{1}{n}\sum_{i=n+1}^{2n}(y_i - \hat{y}_{i,n})^2}, \tag{22}$$

where $\hat{y}_{i,n}$ are the predictions from fitting the model to the first n observations.

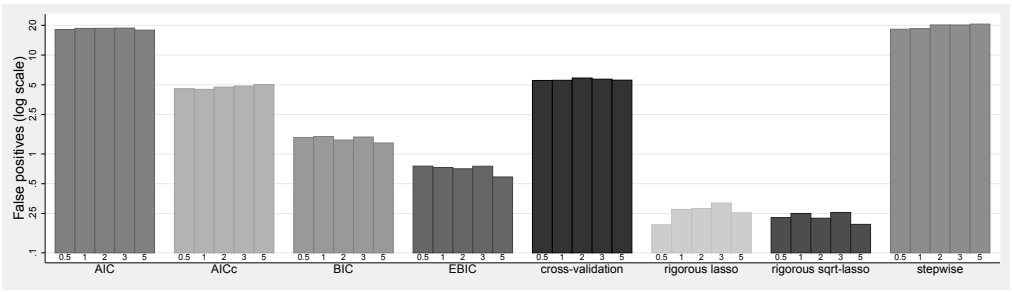
Figure 7 and 8 illustrate results for the following estimation methods implemented in **lassopack**: the lasso with λ selected by AIC, BIC, EBIC$_\xi$ and AIC$_c$ (as implemented in `lasso2`); the rigorous lasso and rigorous square-root lasso (implemented in `rlasso`) using the *X-independent* penalty choice; and the 5-fold cross-validated lasso using the penalty level that minimizes the estimated mean squared prediction error (implemented in `cvlasso`). In addition, we report post-estimation OLS results (denoted by dots).

For comparison, we also show results of stepwise regression (for $p = 100$ only) and the oracle estimator. Stepwise regression starts from the full model and iteratively removes regressors if the p-value is above a pre-defined threshold (10% in our case). Stepwise regression is known to suffer from overfitting and pre-testing bias. However, it still serves as a relevant reference point due to its connection with *ad hoc* model selection using hypothesis testing and the general-to-specific approach. The oracle estimator is OLS applied to the predictors included in the true model. Naturally, the oracle estimator is expected to show the best performance, but is not feasible in practice since the true model is not known. The results of Figure 7 and 8 are also reported in tabular form in Table A.2 and A.3 along with additional information, including results for the rigorous lasso and rigorous square-root lasso with *X-dependent* penalty choice.

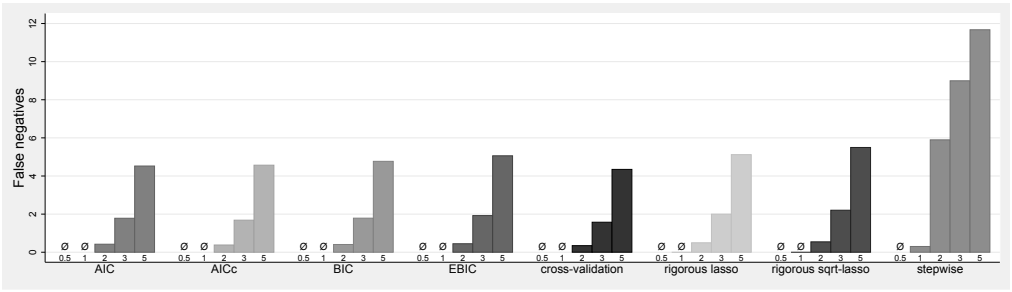
We first summarize the main results in Figure 7 for the case where $p = 100$. AIC and stepwise regression exhibit the worst selection performance, with around 18-20 falsely included predictors on average. While AIC and stepwise regression achieve the lowest RMSE (best in-sample fit), the out-of-sample prediction performance is among the worst—a typical symptom of over-fitting. It is interesting to note that in-sample fit of AIC and stepwise even exceed the fit of the oracle estimator. The corrected AIC improves upon the standard AIC both in terms of false positive rate and prediction performance.

Compared to AIC$_c$, the BIC-type information criteria show similar out-of-sample prediction and better selection performance. While the EBIC performs only marginally better than BIC in terms of false positives, we expect the relative performance of BIC and EBIC to shift in favour of EBIC as p increases relative to n. 5-fold CV with the lasso behaves very similarly to the AIC$_c$ across all measures. The rigorous lasso, rigorous square-root lasso and EBIC exhibit overall the lowest false positive rates, whereas rigorous methods yield slightly higher RMSE and RMSPE than IC and CV-based methods.

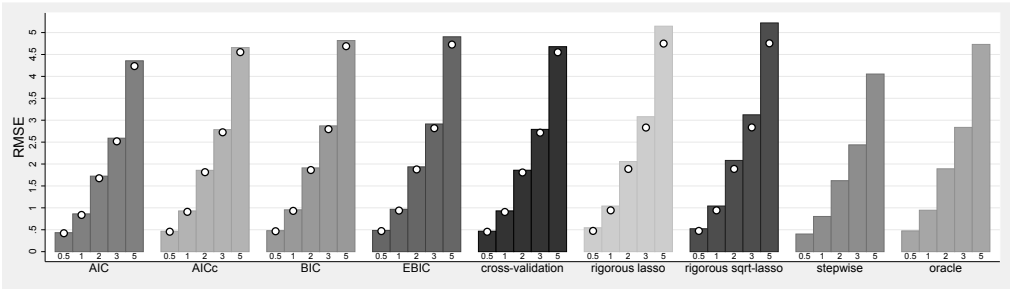
Post-estimation OLS generally exhibits better in-sample fit, indicating that the



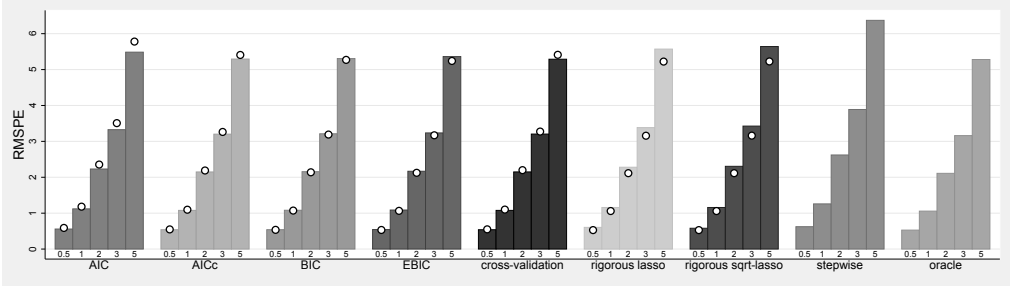
(a) False positive (the number of falsely included predictors) on logarithmic scale



(b) False negative (the number of falsely excluded predictors)



(c) Root mean squared error (in-sample fit)



(d) Root mean squared prediction error (out-of-sample prediction performance)

Notes: Results are shown for noise level $\sigma = \{0.5, 1, 2, 3, 5\}$. ∅ denotes zero false negatives (after rounding to two decimal digits). Post-estimation OLS results are indicated by dots ('∘') if applicable. Cross-validation results are for 5-fold cross-validated lasso. The oracle estimator applies OLS to all predictors in the true model, implying that false positive and false negative frequency are zero by construction. The number of replications is 1,000.

Figure 7: Monte Carlo simulation for an exactly sparse parameter vector with $p = 100$ and $n = 200$.

Method	*Call*	*Seconds*	
		$p = 100$	$p = 220$
Rigorous lasso	`rlasso y x`	0.09	0.24
with X-dependent penalty	`rlasso y x, xdep`	5.92	12.73
Rigorous square-root lasso	`rlasso y x, sqrt`	0.39	0.74
with X-dependent penalty	`rlasso y x, sqrt xdep`	3.34	7.03
Cross-validation	`cvlasso y x, nfolds(5) lopt`	23.50	293.93
Information criteria	`lasso2 y x`	3.06	44.06
Stepwise regression	`stepwise, pr(.1): reg y x`	4.65	–

PC specification: Intel Core i5-6500 with 16GB RAM, Windows 7.

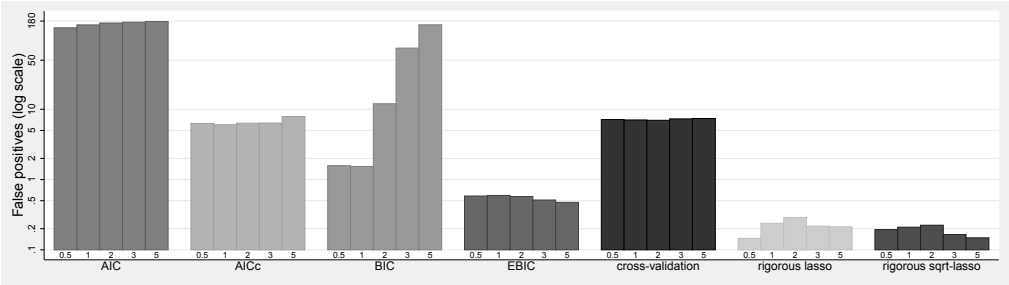
Table 3: Run time with $p = 100$ and $p = 220$.

shrinkage bias from penalization can be alleviated by applying OLS to the selected model. In terms of out-of-sample prediction, post-estimation OLS performs worse than first-step lasso in cases where variable selection performance is poor (in particular for the AIC), but leads to performance improvements for rigorous methods.

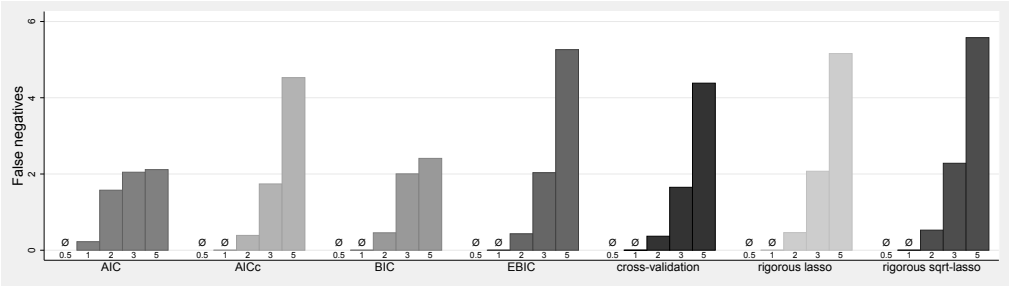
We also present simulation results for the high-dimensional setting in Figure 8. Specifically, we consider $p = 220$ instead of $p = 100$, while keeping the estimation sample size constant at $n = 200$. With on average between 164 and 195 included predictors, the AIC suffers from overfitting. The RMSPE of the AIC exceeds the RMSE by a factor of 5 or more. In comparison, AIC_c and 5-fold cross-validation perform better as model selectors, with a false positive frequency between 6 and 8 predictors.

Despite the large number of predictors to choose from, EBIC and rigorous methods perform generally well in recovering the true structure. The false positive frequency is below 1 across all noise levels, and the false negative rate is zero if σ is 1 or smaller. While the BIC performs similarly to the EBIC for $\sigma = 0.5$ and $\sigma = 1$, its performance resembles the poor performance of AIC for larger noise levels. The Monte Carlo results in Table A.3 highlight that EBIC and rigorous methods are well-suited for the high-dimensional setting where $p > n$, while AIC and BIC are not appropriate.

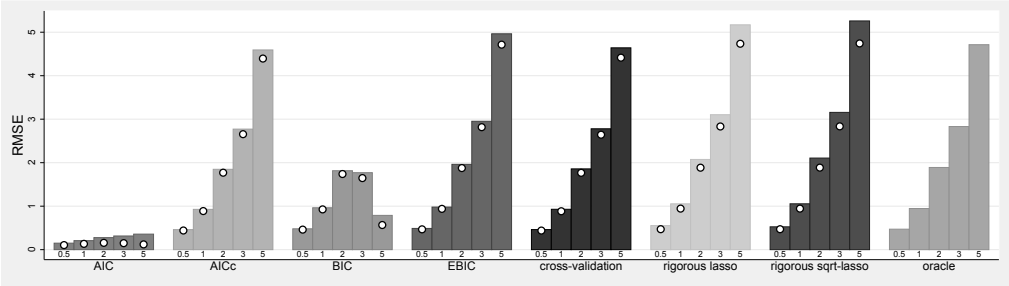
The computational costs of each method are reported in Table 3. `rlasso` with *X-independent* penalty is the fastest method considered. The run-time of lasso and square-root lasso with $p = 100$ is 0.1s and 0.4s, respectively. The computational cost increased only slightly if p is increased to $p = 220$. `rlasso` with *X-dependent* penalty simulates the distribution of the maximum value of the score vector. This process increases the computational cost of the rigorous lasso to 5.9s for $p = 100$ (12.7s for $p = 220$). With an average run-time of 3.1 seconds, `lasso2` is slightly faster than `rlasso` with *X-dependent* penalty if $p = 100$, but slower in the high-dimensional set-up. Unsurprisingly, K-fold cross-validation is the slowest method as it requires the model to be estimated K times for a range of tuning parameters.



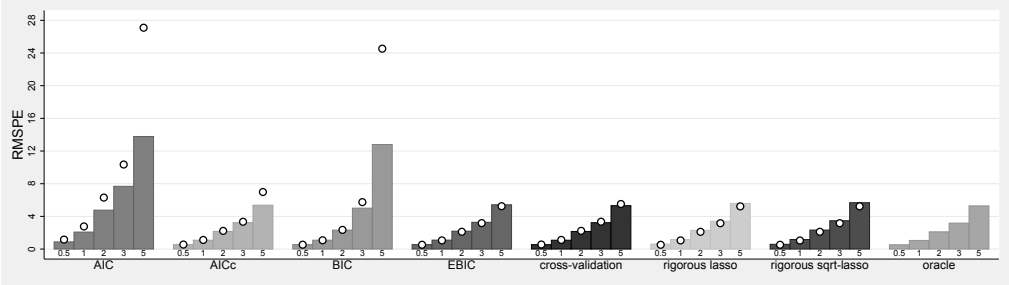
(a) False positive (the number of falsely included predictors) on logarithmic scale



(b) False negative (the number of falsely excluded predictors)



(c) Root mean squared error (in-sample fit)



(d) Root mean squared prediction error (out-of-sample prediction performance)

Notes: Stepwise regression is not reported, as it is infeasible if $p > n$. See also notes in Table 7.

Figure 8: Monte Carlo simulation for an exactly sparse parameter vector with $p = 220$ and $n = 200$.

9 Technical notes

9.1 Pathwise coordinate descent algorithms

`lassopack` implements the elastic net and square-root lasso using coordinate descent algorithms. The algorithm—then referred to as "shooting"—was first proposed by Fu (1998) for the lasso, and by Van der Kooij (2007) for the elastic net. Belloni et al. (2011) and Belloni et al. (2014b) employ the coordinate descent for the square-root lasso, and have kindly provided Matlab code.

Coordinate descent algorithms repeatedly cycle over predictors $j = 1, ..., p$ and update single coefficient estimates until convergence. Suppose the predictors are centered, standardized to have unit variance and the penalty loadings are $\psi_j = 1$ for all j. In that case, the update for coefficient j is obtained using univariate regression of the current partial residuals (i.e., excluding the contribution of predictor j) against predictor j. More precisely, the update for the elastic net is calculated as

$$\tilde{\beta}_j \leftarrow \frac{\mathcal{S}\left(\sum_{i=1}^{n} x_{ij}(y_i - \tilde{y}_i^{(j)}), \lambda\alpha\right)}{1 + \lambda(1-\alpha)}.$$

where $\tilde{\beta}_j$ denotes the current coefficient estimate, $\tilde{y}_i^{(j)} = \sum_{\ell \neq j} x_{i\ell}\tilde{\beta}_\ell$ is the predicted value without the contribution of predictor j. Thus, since the predictors are standardized, $\sum_i x_{ij}(y_i - \tilde{y}_i^{(j)})$ is the OLS estimate of regressing predictor j against the partial residual $(y_i - \tilde{y}_i^{(j)})$. The function $\mathcal{S}(a, b)$, referred to as *soft-tresholding operator*,

$$\mathcal{S}(a,b) = \begin{cases} a - b & \text{if } a > 0 \text{ and } b < |a| \\ a + b & \text{if } a < 0 \text{ and } b < |a| \\ 0 & \text{if } b > |a| \end{cases}$$

sets some of the coefficients equal to zero. The coordinate descent algorithm is spelled out for the square-root lasso in Belloni et al. (2014b, Supplementary Material).[25]

The algorithm requires an initial beta estimate for which the Ridge estimate is used. If the coefficient path is obtained for a list of λ values, `lasso2` starts from the largest λ value and uses previous estimates as initial values ('warm starts'). See Friedman et al. (2007, 2010), and references therein, for further information.

9.2 Standardization

Since penalized regression methods are not invariant to scale, it is common practice to standardize the regressors x_{ij} such that $\sum_i x_{ij}^2 = 1$ before computing the estimation results and then to un-standardize the coefficients after estimation. We refer to this approach as *pre-estimation standardization*. An alternative is to standardize *on the fly* by adapting the penalty loadings. The results are equivalent in theory. In the case

25. Alexandre Belloni provides MATLAB code that implements the pathwise coordinate descent for the square-root lasso, which we have used for comparison.

of the lasso, setting $\psi_j = (\sum_i x_{ij}^2)^{1/2}$ yields the same results as dividing the data by $\sum_i x_{ij}^2$ before estimation. Standardization on-the-fly is the default in `lassopack` as it tends to be faster. Pre-estimation standardization can be employed using the `prestd` option. The `prestd` option can lead to improved numerical precision or more stable results in the case of difficult problems; the cost is (a typically small) computation time required to standardize the data. The `unitloadings` option can be used if the researcher does not want to standardize data. In case the pre-estimation-standardization and standardization-*on-the-fly* results differ, the user can compare the values of the penalized minimized objective function saved in `e(pmse)` (the penalized MSE, for the elastic net) or `e(prmse)` (the penalized root MSE, for the sqrt-lasso).

9.3 Zero-penalization and partialling out

In many applications, theory suggests that specific predictors have an effect on the outcome variable. Hence, it might be desirable to always include these predictors in the model in order to improve finite sample performance. Typical examples are the intercept, a time trend or any other predictor for which the researcher has prior knowledge. `lassopack` offers two approaches for such situations:

- *Zero-penalization:* The `notpen(`*varlist*`)` option of `lasso2` and `cvlasso` allow one to set the penalty for specific predictors to zero, i.e., $\psi_\ell = 0$ for some $\ell \in \{1, \dots, p\}$. Those variables are not subject to penalization and will always be included in the model. `rlasso` supports zero-penalization through the `pnotpen(`*varlist*`)` option which accommodates zero-penalization in the rigorous lasso penalty loadings; see below.
- *Partialling out:* We can also apply the penalized regression method to the data after the effect of certain regressors has been partialled out. Partialling out is supported by `lasso2`, `cvlasso` and `rlasso` using `partial(`*varlist*`)` option. The penalized regression does not yield estimates of the partialled out coefficients directly. Instead, `lassopack` recovers the partialled-out coefficients by post-estimation OLS.

It turns out that the two methods—zero-penalization and partialling out—are numerically equivalent. Formally, suppose we do not want to subject predictors ℓ with $\bar{p} > \ell \geq p$ to penalization. The zero-penalization and partialled-out lasso estimates are defined respectively as

$$\hat{\boldsymbol{\beta}}(\lambda) = \arg\min \frac{1}{n} \sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{\bar{p}} x_{ij}\beta_j - \sum_{\ell=\bar{p}+1}^{p} x_{i\ell}\beta_\ell \right)^2 + \frac{\lambda}{n} \sum_{j=1}^{\bar{p}} \psi_j |\beta_j| \tag{23}$$

and

$$\tilde{\boldsymbol{\beta}}(\lambda) = \arg\min \frac{1}{n} \sum_{i=1}^{n} \left(\tilde{y}_i - \sum_{j=1}^{\bar{p}} \tilde{x}_{ij}\beta_j \right)^2 + \frac{\lambda}{n} \sum_{j=1}^{\bar{p}} \psi_j |\beta_j| \tag{24}$$

where $\tilde{y}_i = y_i - \sum_{\ell=\bar{p}+1}^{p} x_{i\ell}\hat{\delta}_{y,\ell}$ and $\tilde{x}_{ij} = x_{ij} - \sum_{\ell=\bar{p}+1}^{p} x_{i\ell}\hat{\delta}_{j,\ell}$ are the residuals of regressing y and the penalized regressors against the set of unpenalized regressors. The

equivalence states that $\hat{\beta}_j = \tilde{\beta}_j$ for all $j = 1, \ldots, \bar{p}$. The result is spelled out in Yamada (2017) for the lasso and ridge, but holds for the elastic net more generally.

Either the `partial(`*varlist*`)` option or the `notpen(`*varlist*`)` option can be used for variables that should not be penalized by the lasso. The options are equivalent in theory (see above), but numerical results can differ in practice because of the different calculation methods used. Partialling-out variables can lead to improved numerical precision or more stable results in the case of difficult problems compared to zero-penalization, but may be slower in terms of computation time.

The estimation of penalty loadings in the rigorous lasso introduces an additional complication that necessitates the `rlasso`-specific option `pnotppen(`*varlist*`)`. The theory for the `rlasso` penalty loadings is based on the penalized regressors after partialling out the unpenalized variables. The `pnotpen(`*varlist*`)` guarantees that the penalty loadings for the penalized regressors are the same as if the unpenalized regressors had instead first been partialled-out.

The `fe` fixed-effects option is equivalent to (but computationally faster and more accurate than) specifying unpenalized panel-specific dummies. The fixed-effects ('within') transformation also removes the constant as well as the fixed effects. The panel variable used by the `fe` option is the panel variable set by `xtset`. If installed, the within transformation uses the fast **ftools** package by Correia (2016).

The `prestd` option, as well as the `notpen(`*varlist*`)` and `pnotpen(`*varlist*`)` options, can be used as simple checks for numerical stability by comparing results that should be equivalent in theory. The values of the penalized minimized objective function saved in `e(pmse)` for the elastic net and `e(prmse)` for the square-root lasso may also be used for comparison.

9.4 Treatment of the constant

By default the constant, if present, is not penalized; this is equivalent to mean-centering prior to estimation. The `partial(`*varlist*`)` option also partials out the constant (if present). To partial out the constant only, we can specify `partial(_cons)`. Both `partial(`*varlist*`)` and `fe` mean-center the data; the `noconstant` option is redundant in this case and may not be specified with these options. If the `noconstant` option is specified an intercept is not included in the model, but the estimated penalty loadings are still estimated using mean-centered regressors (see the `center` option).

10 Acknowledgments

We thank Alexandre Belloni, who has provided MATLAB code for the square-root lasso, and Sergio Correia for supporting us with the use of **ftools**. We also thank Christopher F Baum, Jan Ditzen, David M Drukker, Martin Spindler, as well as participants of the 2018 London Stata Conference and the 2018 Swiss Stata Users Group meeting for many helpful comments and suggestions. We have received excellent comments from an

anonymous referee which have improved the article. We are also grateful to the many users that have provided feedback. All remaining errors are our own.

11 References

Ahrens, A., C. B. Hansen, and M. E. Schaffer. 2018. PDSLASSO: Stata module for post-selection and post-regularization OLS or IV estimation and inference. Statistical Software Components, Boston College Department of Economics. https://ideas.repec.org/c/boc/bocode/s458459.html.

Akaike, H. 1974. A new look at the statistical model identification. *IEEE Transactions on Automatic Control* 19(6): 716–723.

Andrews, D. W. K. 1991. Asymptotic optimality of generalized CL, cross-validation, and generalized cross-validation in regression with heteroskedastic errors. *Journal of Econometrics* 47(2): 359–377. http://www.sciencedirect.com/science/article/pii/030440769190107O.

Arlot, S., and A. Celisse. 2010. A survey of cross-validation procedures for model selection. *Statist. Surv.* 4: 40–79. https://doi.org/10.1214/09-SS054.

Athey, S. 2017. The Impact of Machine Learning on Economics. https://www.nber.org/chapters/c14009.pdf.

Athey, S., and G. Imbens. 2019. Machine Learning Methods Economists Should Know About . https://arxiv.org/abs/1903.10075.

Belloni, A., D. Chen, V. Chernozhukov, and C. Hansen. 2012. Sparse Models and Methods for Optimal Instruments With an Application to Eminent Domain. *Econometrica* 80(6): 2369–2429. http://dx.doi.org/10.3982/ECTA9626.

Belloni, A., and V. Chernozhukov. 2011. High Dimensional Sparse Econometric Models: An Introduction. In *Inverse Problems and High-Dimensional Estimation SE - 3*, ed. P. Alquier, E. Gautier, and G. Stoltz, 121–156. Lecture Notes in Statistics, Springer Berlin Heidelberg.

———. 2013. Least squares after model selection in high-dimensional sparse models. *Bernoulli* 19(2): 521–547. http://dx.doi.org/10.3150/11-BEJ410.

Belloni, A., V. Chernozhukov, and C. Hansen. 2014a. Inference on treatment effects after selection among high-dimensional controls. *Review of Economic Studies* 81: 608–650. https://doi.org/10.1093/restud/rdt044.

Belloni, A., V. Chernozhukov, C. Hansen, and D. Kozbur. 2016. Inference in High Dimensional Panel Models with an Application to Gun Control. *Journal of Business & Economic Statistics* 34(4): 590–605. https://doi.org/10.1080/07350015.2015.1102733.

Belloni, A., V. Chernozhukov, and L. Wang. 2011. Square-root lasso: pivotal recovery of sparse signals via conic programming. *Biometrika* 98(4): 791–806. https://doi.org/10.1093/biomet/asr043.

———. 2014b. Pivotal estimation via square-root Lasso in nonparametric regression. *The Annals of Statistics* 42(2): 757–788. http://dx.doi.org/10.1214/14-AOS1204.

Bergmeir, C., R. J. Hyndman, and B. Koo. 2018. A note on the validity of cross-validation for evaluating autoregressive time series prediction. *Computational Statistics & Data Analysis* 120: 70–83. https://doi.org/10.1016/j.csda.2017.11.003.

Bickel, P. J., Y. Ritov, and A. B. Tsybakov. 2009. Simultaneous Analysis of Lasso and Dantzig Selector. *The Annals of Statistics* 37(4): 1705–1732. http:/doi.org/10.1214/08-AOS620.

Buhlmann, P. 2013. Statistical significance in high-dimensional linear models. *Bernoulli* 19(4): 1212–1242. https://doi.org/10.3150/12-BEJSP11.

Bühlmann, P., and S. Van de Geer. 2011. *Statistics for High-Dimensional Data*. Berlin, Heidelberg: Springer-Verlag.

Burman, P., E. Chow, and D. Nolan. 1994. A cross-validatory method for dependent data. *Biometrika* 81(2): 351–358. http://dx.doi.org/10.1093/biomet/81.2.351.

Carrasco, M. 2012. A regularization approach to the many instruments problem. *Journal of Econometrics* 170: 383–398. https://doi.org/10.1016/j.jeconom.2012.05.012.

Chen, J., and Z. Chen. 2008. Extended Bayesian information criteria for model selection with large model spaces. *Biometrika* 95(3): 759–771. + http://dx.doi.org/10.1093/biomet/asn034.

Chernozhukov, V., D. Chetverikov, and K. Kato. 2013. Gaussian approximations and multiplier bootstrap for maxima of sums of high-dimensional random vectors. *Ann. Statist.* 41(6): 2786–2819. https://doi.org/10.1214/13-AOS1161.

Chernozhukov, V., C. Hansen, and M. Spindler. 2015. Post-Selection and Post-Regularization Inference in Linear Models with Many Controls and Instruments. *American Economic Review* 105(5): 486–490. https://doi.org/10.1257/aer.p20151022.

———. 2016. High-Dimensional Metrics in R. *arXiv preprint arXiv:1603.01700* .

Chetverikov, D., Z. Liao, and V. Chernozhukov. 2019. On cross-validated Lasso . http://arxiv.org/abs/1605.02214.

Correia, S. 2016. FTOOLS: Stata module to provide alternatives to common Stata commands optimized for large datasets. Statistical Software Components, Boston College Department of Economics. https://ideas.repec.org/c/boc/bocode/s458213.html.

Dicker, L. H. 2016. Ridge regression and asymptotic minimax estimation over spheres of growing dimension. *Bernoulli* 22(1): 1–37. https://doi.org/10.3150/14-BEJ609.

Dobriban, E., and S. Wager. 2018. High-dimensional asymptotics of prediction: Ridge regression and classification. *Annals of Statistics* 46(1): 247–279.

Frank, l. E., and J. H. Friedman. 1993. A Statistical View of Some Chemometrics Regression Tools. *Technometrics* 35(2): 109–135.

Friedman, J., T. Hastie, H. Höfling, and R. Tibshirani. 2007. Pathwise coordinate optimization. *The Annals of Applied Statistics* 1(2): 302–332. http://projecteuclid.org/euclid.aoas/1196438020.

Friedman, J., T. Hastie, and R. Tibshirani. 2010. Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software* 33(1): 1–22. http://www.jstatsoft.org/v33/i01/.

Fu, W. J. 1998. Penalized Regressions: The Bridge Versus the Lasso. *Journal of Computational and Graphical Statistics* 7(3): 397–416.

Geisser, S. 1975. The Predictive Sample Reuse Method with Applications. *Journal of the American Statistical Association* 70(350): 320–328.

Hansen, C., and D. Kozbur. 2014. Instrumental variables estimation with many weak instruments using regularized JIVE. *Journal of Econometrics* 182(2): 290–308.

Hastie, T., R. Tibshirani, and J. Friedman. 2009. *The Elements of Statistical Learning*. 2nd ed. New York: Springer-Verlag.

Hastie, T., R. Tibshirani, and M. J. Wainwright. 2015. *Statistical Learning with Sparsity: The Lasso and Generalizations*. Monographs on Statistics & Applied Probability, Boca Raton: CRC Press, Taylor & Francis.

Hoerl, A. E., and R. W. Kennard. 1970. Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics* 12(1): 55–67.

Hsu, D., S. M. Kakade, and T. Zhang. 2014. Random Design Analysis of Ridge Regression. *Foundations of Computational Mathematics* 14(3): 569–600. https://doi.org/10.1007/s10208-014-9192-1.

Huang, J., S. Ma, and C.-H. Zhang. 2008. Adaptive Lasso for Sparse High-Dimensional Regression Models. *Statistica Sinica* 18(4): 1603–1618. http://www.jstor.org/stable/24308572.

Hurvich, C. M., and C.-L. Tsai. 1989. Regression and time series model selection in small samples. *Biometrika* 76(2): 297–307. http://dx.doi.org/10.1093/biomet/76.2.297.

Hyndman, Rob, J., and G. Athanasopoulos. 2018. *Forecasting: Principles and Practice*. 2nd ed. https://otexts.com/fpp2/.

Jing, B.-Y., Q.-M. Shao, and Q. Wang. 2003. Self-normalized Cramér-type large deviations for independent random variables. *The Annals of Probability* 31(4): 2167–2215. http://dx.doi.org/10.1214/aop/1068646382.

Kleinberg, J., H. Lakkaraju, J. Leskovec, J. Ludwig, and S. Mullainathan. 2018. Human Decisions and Machine Predictions. *The Quarterly Journal of Economics* 133(1): 237–293. http://dx.doi.org/10.1093/qje/qjx032.

Lockhart, R., J. Taylor, R. J. Tibshirani, and R. Tibshirani. 2014. A Significance Test for the Lasso. *Annals of Statistics* 42(2): 413–468. https://doi.org/10.1214/13-AOS1175.

Meinshausen, N., and P. Bühlmann. 2006. High-dimensional graphs and variable selection with the Lasso. *The Annals of Statistics* 34(3): 1436–1462. https://doi.org/10.1214/009053606000000281.

Meinshausen, N., L. Meier, and P. Bühlmann. 2009. p-Values for High-Dimensional Regression. *Journal of the American Statistical Association* 104(488): 1671–1681.

Mullainathan, S., and J. Spiess. 2017. Machine Learning: An Applied Econometric Approach. *Journal of Economic Perspectives* 31(2): 87–106. http://www.aeaweb.org/articles?id=10.1257/jep.31.2.87.

Raninen, E., and E. Ollila. 2017. Scaled and square-root elastic net. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 4336–4340.

Schwarz, G. 1978. Estimating the Dimension of a Model. *The Annals of Statistics* 6(2): 461–464.

Shao, J. 1993. Linear Model Selection by Cross-Validation. *Journal of the American Statistical Association* 88(422): 486–494. http://www.jstor.org/stable/2290328.

———. 1997. An asymptotic theory for linear model selection. *Statistica Sinica* 7: 221–264.

Stone, M. 1977. An Asymptotic Equivalence of Choice of Model by Cross-Validation and Akaike's Criterion. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 39(1): 44–47. https://www.jstor.org/stable/2984877.

Sugiura, N. 1978. Further analysts of the data by akaike' s information criterion and the finite corrections. *Communications in Statistics - Theory and Methods* 7(1): 13–26. https://doi.org/10.1080/03610927808827599.

Tibshirani, R. 1996. Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)* 58(1): 267–288. http://www.jstor.org/stable/2346178.

Tibshirani, R. J., and J. Taylor. 2012. Degrees of freedom in lasso problems. *Annals of Statistics* .

Tikhonov, A. N. 1963. On the solution of ill-posed problems and the method of regularization. In *Doklady Akademii Nauk*, vol. 151, 501–504. Russian Academy of Sciences.

Varian, H. R. 2014. Big Data: New Tricks for Econometrics. *The Journal of Economic Perspectives* 28(2): pp. 3–27. http://www.jstor.org/stable/23723482.

Wasserman, L., and K. Roeder. 2009. High-dimensional variable selection. *Annals of Statistics* 37(5A): 2178–2201. http://dx.doi.org/10.1214/08-AOS646.

Weilenmann, B., I. Seidl, and T. Schulz. 2017. The socio-economic determinants of urban sprawl between 1980 and 2010 in Switzerland. *Landscape and Urban Planning* 157: 468–482.

Yamada, H. 2017. The FrischWaughLovell theorem for the lasso and the ridge regression. *Communications in Statistics - Theory and Methods* 46(21): 10897–10902. http://dx.doi.org/10.1080/03610926.2016.1252403.

Yang, Y. 2005. Can the strengths of AIC and BIC be shared? A conflict between model indentification and regression estimation. *Biometrika* 92(4): 937–950.

———. 2006. Comparing learning methods for classification. *Statistica Sinica* 16(2): 635–657. https://www.jstor.org/stable/24307562.

Zhang, Y., R. Li, and C.-L. Tsai. 2010. Regularization Parameter Selections via Generalized Information Criterion. *Journal of the American Statistical Association* 105(489): 312–323. https://doi.org/10.1198/jasa.2009.tm08013.

Zhao, P., and B. Yu. 2006. On Model Selection Consistency of Lasso. *Journal of Machine Learning Research* 7: 2541–2563. http://dl.acm.org/citation.cfm?id=1248547.1248637.

Zou, H. 2006. The Adaptive Lasso and Its Oracle Properties. *Journal of the American Statistical Association* 101(476): 1418–1429.

Zou, H., and T. Hastie. 2005. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society. Series B: Statistical Methodology* 67(2): 301–320.

Zou, H., T. Hastie, and R. Tibshirani. 2007. On the "degrees of freedom" of the lasso. *Ann. Statist.* 35(5): 2173–2192. https://doi.org/10.1214/009053607000000127.

Zou, H., and H. H. Zhang. 2009. On the adaptive elastic-net with a diverging number of parameters. *Ann. Statist.* 37(4): 1733–1751. https://doi.org/10.1214/08-AOS625.

About the authors

Achim Ahrens is Post-doctoral Research Fellow at The Economic and Social Research Institute in Dublin, Ireland.

Mark E. Schaffer is Professor of Econonomics in the School of Social Sciences at Heriot-Watt University, Edinburgh, UK, and a Research Fellow at the Centre for Economic Policy Research (CEPR), London and the Institute for the Study of Labour (IZA), Bonn.

Christian B. Hansen is the Wallace W. Booth Professor of Econometrics and Statistics at the University of Chicago Booth School of Business.

A Appendix

A.1 Additional Monte Carlo results

In this supplementary section, we consider an additional design. Instead of defining β_j as either 0 or +1, we let the non-zero coefficients alternate between +1 and -1. That is, we define the sparse parameter vector as $\beta_j = (-1)^j.\mathbb{1}\{j \leq s\}$ for $j = 1, \ldots, p$ with $s = 20$. All remaining parameters are as in Section 8, and we consider $p = 100$.

The results are reported in Table A.1. Compared to the base specification in Section 8, the model selection performance deteriorates drastically. The false negative rate is high across all methods. When σ is equal to 2 or larger, BIC-type information criteria and rigorous methods often select no variables, whereas AIC and stepwise regression tend to overselect.

On the other hand, out-of-sample prediction can still be satisfactory despite the poor selection performance. For example, at $\sigma = 2$, the RMSPE of cross-validation is only 9.0% above the RMSPE of the oracle estimator (2.3 compared to 2.11), even though only 4.2 predictors are correctly selected on average. The Monte Carlo results highlight an important insight: model selection is generally a difficult task. Yet, satisfactory prediction can be achieved without perfect model selection.

	σ	`lasso2` AIC	AIC_c	BIC	$EBIC_\xi$	`cvlasso`	`rlasso` lasso	`xdep`	$\sqrt{\text{lasso}}$	`xdep`	Step wise	Oracle
$\hat{s}$	.5	77.92	57.07	51.41	4.74	65.90	2.34	2.44	2.03	2.19	37.31	–
	1	77.80	51.69	3.90	1.88	60.95	1.65	1.83	1.26	1.51	37.30	–
	2	56.31	11.76	1.21	0.28	12.12	0.31	0.41	0.20	0.32	31.68	–
	3	26.70	6.40	0.35	0.05	5.28	0.06	0.09	0.04	0.06	27.79	–
	5	15.06	4.02	0.12	0.01	3.23	0.01	0.03	0.00	0.02	25.03	–
False pos.	.5	57.92	37.07	31.44	1.58	45.90	0.32	0.35	0.25	0.27	18.31	–
	1	57.85	33.25	1.40	0.59	41.90	0.37	0.36	0.35	0.32	18.65	–
	2	41.91	7.50	0.91	0.87	7.95	0.80	0.74	0.86	0.78	19.76	–
	3	20.08	4.43	0.94	0.97	3.85	0.96	0.95	0.97	0.96	19.88	–
	5	11.77	3.21	0.99	1.00	2.78	0.99	0.99	1.00	0.99	19.68	–
False neg.	.5	0.00	0.00	0.03	16.80	0.00	17.98	17.91	18.20	18.07	0.00	–
	1	0.05	1.56	17.45	18.46	0.95	18.61	18.47	18.91	18.71	0.35	–
	2	5.59	15.71	19.19	19.77	15.76	19.75	19.67	19.84	19.74	7.08	–
	3	13.29	17.88	19.79	19.96	18.26	19.96	19.94	19.97	19.96	11.08	–
	5	16.45	18.82	19.95	20.00	19.07	19.99	19.99	20.00	19.99	13.65	–
RMSE	.5	0.373	0.434	0.464	1.108	0.409	1.208	1.199	1.230	1.216	0.402	0.474
		(*0.359*)	(*0.386*)	(*0.399*)	(*1.063*)	(*0.372*)	(*1.108*)	(*1.105*)	(*1.119*)	(*1.112*)	(–)	(–)
	1	0.743	0.901	1.418	1.466	0.849	1.519	1.510	1.534	1.524	0.801	0.944
		(*0.715*)	(*0.807*)	(*1.375*)	(*1.429*)	(*0.769*)	(*1.429*)	(*1.420*)	(*1.452*)	(*1.436*)	(–)	(–)
	2	1.646	2.110	2.280	2.316	2.126	2.328	2.326	2.331	2.328	1.629	1.890
		(*1.577*)	(*2.041*)	(*2.255*)	(*2.307*)	(*2.051*)	(*2.302*)	(*2.294*)	(*2.312*)	(*2.300*)	(–)	(–)
	3	2.770	3.064	3.199	3.214	3.103	3.218	3.217	3.218	3.217	2.457	2.835
		(*2.690*)	(*3.003*)	(*3.188*)	(*3.212*)	(*3.043*)	(*3.210*)	(*3.207*)	(*3.213*)	(*3.210*)	(–)	(–)
	5	4.731	4.988	5.123	5.131	5.033	5.133	5.132	5.133	5.133	4.099	4.733
		(*4.636*)	(*4.923*)	(*5.117*)	(*5.131*)	(*4.968*)	(*5.130*)	(*5.128*)	(*5.132*)	(*5.129*)	(–)	(–)
RMSPE	.5	0.638	0.622	0.638	1.143	0.615	1.226	1.218	1.247	1.234	0.623	0.527
		(*0.684*)	(*0.633*)	(*0.618*)	(*1.112*)	(*0.654*)	(*1.138*)	(*1.134*)	(*1.152*)	(*1.143*)	(–)	(–)
	1	1.279	1.260	1.468	1.500	1.245	1.542	1.535	1.555	1.546	1.258	1.057
		(*1.369*)	(*1.291*)	(*1.455*)	(*1.482*)	(*1.312*)	(*1.483*)	(*1.473*)	(*1.505*)	(*1.490*)	(–)	(–)
	2	2.436	2.297	2.318	2.336	2.299	2.342	2.341	2.344	2.342	2.595	2.107
		(*2.628*)	(*2.374*)	(*2.324*)	(*2.339*)	(*2.372*)	(*2.340*)	(*2.337*)	(*2.342*)	(*2.339*)	(–)	(–)
	3	3.378	3.234	3.237	3.240	3.233	3.241	3.241	3.241	3.241	3.821	3.163
		(*3.570*)	(*3.322*)	(*3.247*)	(*3.243*)	(*3.303*)	(*3.244*)	(*3.244*)	(*3.243*)	(*3.244*)	(–)	(–)
	5	5.341	5.187	5.162	5.161	5.176	5.161	5.161	5.161	5.161	6.245	5.285
		(*5.577*)	(*5.306*)	(*5.172*)	(*5.162*)	(*5.273*)	(*5.163*)	(*5.164*)	(*5.161*)	(*5.163*)	(–)	(–)

Notes: $\hat{s}$ denotes the number of selected variables excluding the constant. 'False pos.' and 'False neg.' denote the number of falsely included and falsely excluded variables, respectively. 'Bias' is the ℓ_1-norm bias defined as $\sum_j |\hat{\beta}_j - \beta_j|$ for $j = 1, \ldots, p$. 'RMSE' and 'RMSPE' are defined in equation (22). Post-estimation OLS results are shown in parentheses if applicable. `cvlasso` results are for 5-fold cross-validation. The oracle estimator applies OLS to all predictors in the true model, implying that false positive and false negative frequency are zero by construction. The number of replications is 1,000.

Table A.1: Monte Carlo simulation for exactly sparse parameter vector with alternating β_j.

A.2 Detailed Results of Section 8

	σ	`lasso2` AIC	AIC_c	BIC	$EBIC_\xi$	`cvlasso`	`rlasso` lasso	`xdep`	$\sqrt{\text{lasso}}$	`xdep`	Step wise	Oracle
$\hat{s}$	.5	38.14	24.57	21.47	20.75	25.54	20.19	20.22	20.23	20.27	37.26	–
	1	38.62	24.50	21.50	20.73	25.56	20.27	20.30	20.25	20.27	37.23	–
	2	38.22	24.37	20.98	20.26	25.51	19.78	19.83	19.68	19.74	33.27	–
	3	36.94	23.17	19.69	18.83	24.13	18.32	18.39	18.05	18.17	30.15	–
	5	33.35	20.46	16.52	15.52	21.23	15.13	15.25	14.70	14.86	27.90	–
False pos.	.5	18.14	4.57	1.47	0.75	5.54	0.19	0.22	0.23	0.27	18.26	–
	1	18.62	4.50	1.50	0.73	5.56	0.28	0.30	0.25	0.28	18.53	–
	2	18.64	4.75	1.38	0.71	5.86	0.28	0.32	0.22	0.25	20.17	–
	3	18.73	4.86	1.48	0.75	5.71	0.32	0.35	0.26	0.29	20.14	–
	5	17.87	5.03	1.29	0.58	5.58	0.25	0.28	0.19	0.22	20.58	–
False neg.	.5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	–
	1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.30	–
	2	0.42	0.38	0.41	0.45	0.35	0.50	0.48	0.55	0.52	5.90	–
	3	1.78	1.68	1.79	1.92	1.58	2.00	1.96	2.20	2.12	8.99	–
	5	4.52	4.57	4.77	5.06	4.35	5.12	5.03	5.50	5.36	11.67	–
Bias	.5	3.139	2.002	1.910	1.898	1.999	2.074	2.041	1.990	1.972	5.529	1.803
		(*4.155*)	(*2.208*)	(*1.952*)	(*1.894*)	(*2.265*)	(*1.835*)	(*1.838*)	(*1.842*)	(*1.846*)	(–)	(–)
	1	6.421	3.974	3.798	3.771	3.983	3.964	3.931	3.958	3.922	11.379	3.578
		(*8.459*)	(*4.374*)	(*3.891*)	(*3.756*)	(*4.522*)	(*3.670*)	(*3.674*)	(*3.663*)	(*3.667*)	(–)	(–)
	2	12.510	7.836	7.412	7.370	7.847	7.607	7.569	7.670	7.609	28.241	7.117
		(*16.563*)	(*8.741*)	(*7.601*)	(*7.402*)	(*9.015*)	(*7.277*)	(*7.281*)	(*7.283*)	(*7.272*)	(–)	(–)
	3	18.294	11.093	10.461	10.377	11.134	10.501	10.470	10.570	10.518	41.475	10.679
		(*24.274*)	(*12.427*)	(*10.913*)	(*10.641*)	(*12.828*)	(*10.542*)	(*10.530*)	(*10.633*)	(*10.572*)	(–)	(–)
	5	26.965	15.878	14.529	14.371	15.666	14.292	14.277	14.343	14.313	63.995	18.040
		(*36.638*)	(*18.423*)	(*15.558*)	(*15.326*)	(*18.617*)	(*15.337*)	(*15.271*)	(*15.582*)	(*15.442*)	(–)	(–)
RMSE	.5	0.433	0.466	0.479	0.484	0.466	0.546	0.536	0.522	0.517	0.403	0.474
		(*0.421*)	(*0.456*)	(*0.467*)	(*0.470*)	(*0.454*)	(*0.473*)	(*0.473*)	(*0.473*)	(*0.473*)	(–)	(–)
	1	0.862	0.930	0.955	0.967	0.931	1.041	1.031	1.042	1.030	0.803	0.945
		(*0.838*)	(*0.909*)	(*0.931*)	(*0.938*)	(*0.905*)	(*0.943*)	(*0.943*)	(*0.943*)	(*0.943*)	(–)	(–)
	2	1.724	1.856	1.912	1.935	1.859	2.057	2.042	2.082	2.059	1.620	1.891
		(*1.675*)	(*1.815*)	(*1.863*)	(*1.877*)	(*1.807*)	(*1.887*)	(*1.887*)	(*1.888*)	(*1.888*)	(–)	(–)
	3	2.589	2.785	2.871	2.914	2.791	3.080	3.059	3.123	3.089	2.437	2.836
		(*2.518*)	(*2.725*)	(*2.796*)	(*2.818*)	(*2.716*)	(*2.833*)	(*2.832*)	(*2.836*)	(*2.834*)	(–)	(–)
	5	4.356	4.659	4.819	4.904	4.678	5.146	5.113	5.220	5.165	4.054	4.730
		(*4.236*)	(*4.554*)	(*4.690*)	(*4.727*)	(*4.551*)	(*4.749*)	(*4.747*)	(*4.756*)	(*4.752*)	(–)	(–)
RMSPE	.5	0.558	0.539	0.540	0.543	0.539	0.605	0.594	0.580	0.574	0.623	0.528
		(*0.589*)	(*0.548*)	(*0.536*)	(*0.533*)	(*0.550*)	(*0.529*)	(*0.529*)	(*0.529*)	(*0.529*)	(–)	(–)
	1	1.120	1.078	1.081	1.087	1.078	1.158	1.148	1.159	1.147	1.259	1.057
		(*1.181*)	(*1.096*)	(*1.073*)	(*1.065*)	(*1.100*)	(*1.060*)	(*1.060*)	(*1.059*)	(*1.060*)	(–)	(–)
	2	2.231	2.149	2.155	2.168	2.149	2.280	2.265	2.305	2.282	2.621	2.110
		(*2.355*)	(*2.189*)	(*2.139*)	(*2.125*)	(*2.199*)	(*2.115*)	(*2.115*)	(*2.115*)	(*2.114*)	(–)	(–)
	3	3.325	3.201	3.211	3.235	3.203	3.384	3.364	3.426	3.393	3.888	3.161
		(*3.509*)	(*3.263*)	(*3.191*)	(*3.170*)	(*3.274*)	(*3.158*)	(*3.158*)	(*3.160*)	(*3.159*)	(–)	(–)
	5	5.485	5.293	5.307	5.361	5.289	5.571	5.540	5.642	5.590	6.372	5.280
		(*5.781*)	(*5.407*)	(*5.271*)	(*5.241*)	(*5.412*)	(*5.223*)	(*5.224*)	(*5.227*)	(*5.225*)	(–)	(–)

Notes: See notes in Table A.1.

Table A.2: Monte Carlo simulation for an exactly sparse parameter vector with $p = 100$ and $n = 200$.

	σ	`lasso2` AIC	AIC_c	BIC	$EBIC_\xi$	`cvlasso`	`rlasso` lasso	`rlasso` `xdep`	$\sqrt{\text{lasso}}$	$\sqrt{\text{lasso}}$ `xdep`	Oracle
$\hat{s}$	.5	164.38	26.29	21.58	20.58	27.16	20.15	20.17	20.19	20.22	–
	1	178.68	26.03	21.53	20.59	27.05	20.24	20.26	20.21	20.24	–
	2	187.55	25.95	31.54	20.14	26.61	19.83	19.87	19.70	19.76	–
	3	191.44	24.64	92.26	18.48	25.65	18.14	18.20	17.88	17.98	–
	5	195.18	23.37	177.00	15.21	23.02	15.05	15.14	14.57	14.73	–
False pos.	.5	144.38	6.29	1.58	0.58	7.16	0.15	0.17	0.19	0.22	–
	1	158.91	6.03	1.54	0.59	7.06	0.24	0.26	0.21	0.24	–
	2	169.13	6.34	12.00	0.57	6.97	0.29	0.31	0.22	0.26	–
	3	173.49	6.37	74.26	0.51	7.30	0.22	0.23	0.16	0.20	–
	5	177.29	7.90	159.41	0.47	7.40	0.21	0.25	0.15	0.18	–
False neg.	.5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	–
	1	0.22	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	–
	2	1.58	0.39	0.46	0.43	0.37	0.46	0.45	0.53	0.50	–
	3	2.05	1.74	2.00	2.04	1.65	2.07	2.04	2.28	2.21	–
	5	2.12	4.53	2.41	5.26	4.38	5.16	5.11	5.58	5.45	–
Bias	.5	19.205	2.019	1.913	1.904	2.014	2.099	2.070	2.001	1.982	1.793
		(*30.730*)	(*2.338*)	(*1.975*)	(*1.876*)	(*2.388*)	(*1.821*)	(*1.825*)	(*1.829*)	(*1.832*)	(–)
	1	53.073	4.029	3.827	3.807	4.030	4.007	3.978	4.002	3.965	3.590
		(*79.744*)	(*4.663*)	(*3.960*)	(*3.763*)	(*4.775*)	(*3.673*)	(*3.678*)	(*3.665*)	(*3.673*)	(–)
	2	133.674	7.980	14.932	7.512	7.901	7.751	7.716	7.827	7.766	7.190
		(*191.444*)	(*9.239*)	(*17.226*)	(*7.470*)	(*9.320*)	(*7.376*)	(*7.374*)	(*7.382*)	(*7.375*)	(–)
	3	220.488	11.341	95.127	10.370	11.071	10.539	10.508	10.629	10.571	10.813
		(*318.761*)	(*13.494*)	(*123.928*)	(*10.610*)	(*13.410*)	(*10.519*)	(*10.499*)	(*10.631*)	(*10.587*)	(–)
	5	409.487	18.294	365.813	14.339	15.709	14.299	14.283	14.367	14.331	18.060
		(*871.303*)	(*68.788*)	(*771.102*)	(*15.526*)	(*19.627*)	(*15.372*)	(*15.325*)	(*15.635*)	(*15.503*)	(–)
RMSE	.5	0.150	0.460	0.480	0.488	0.462	0.553	0.544	0.525	0.519	0.471
		(*0.105*)	(*0.441*)	(*0.461*)	(*0.467*)	(*0.439*)	(*0.471*)	(*0.470*)	(*0.470*)	(*0.470*)	(–)
	1	0.207	0.927	0.963	0.980	0.929	1.054	1.044	1.055	1.043	0.946
		(*0.134*)	(*0.888*)	(*0.926*)	(*0.939*)	(*0.884*)	(*0.944*)	(*0.944*)	(*0.945*)	(*0.944*)	(–)
	2	0.277	1.848	1.816	1.963	1.857	2.075	2.061	2.106	2.082	1.890
		(*0.157*)	(*1.770*)	(*1.739*)	(*1.876*)	(*1.767*)	(*1.885*)	(*1.885*)	(*1.887*)	(*1.886*)	(–)
	3	0.314	2.772	1.770	2.952	2.780	3.103	3.084	3.156	3.122	2.831
		(*0.150*)	(*2.656*)	(*1.646*)	(*2.817*)	(*2.643*)	(*2.832*)	(*2.832*)	(*2.835*)	(*2.834*)	(–)
	5	0.357	4.592	0.788	4.964	4.641	5.170	5.140	5.260	5.203	4.713
		(*0.122*)	(*4.395*)	(*0.567*)	(*4.714*)	(*4.415*)	(*4.735*)	(*4.733*)	(*4.743*)	(*4.739*)	(–)
RMSPE	.5	0.875	0.541	0.544	0.549	0.542	0.614	0.604	0.584	0.578	0.527
		(*1.165*)	(*0.559*)	(*0.539*)	(*0.532*)	(*0.561*)	(*0.528*)	(*0.528*)	(*0.528*)	(*0.528*)	(–)
	1	2.079	1.083	1.088	1.097	1.084	1.169	1.159	1.170	1.157	1.056
		(*2.773*)	(*1.118*)	(*1.078*)	(*1.064*)	(*1.122*)	(*1.058*)	(*1.058*)	(*1.057*)	(*1.058*)	(–)
	2	4.765	2.159	2.320	2.190	2.158	2.296	2.282	2.327	2.303	2.109
		(*6.304*)	(*2.228*)	(*2.352*)	(*2.121*)	(*2.230*)	(*2.114*)	(*2.114*)	(*2.114*)	(*2.114*)	(–)
	3	7.683	3.232	5.010	3.280	3.227	3.414	3.396	3.466	3.432	3.174
		(*10.352*)	(*3.344*)	(*5.749*)	(*3.180*)	(*3.349*)	(*3.167*)	(*3.166*)	(*3.169*)	(*3.168*)	(–)
	5	13.782	5.369	12.799	5.422	5.315	5.593	5.565	5.678	5.625	5.285
		(*27.099*)	(*6.988*)	(*24.532*)	(*5.249*)	(*5.515*)	(*5.227*)	(*5.227*)	(*5.231*)	(*5.229*)	(–)

Notes: Stepwise regression is not reported, as it is infeasible if $p > n$. See also notes in Table A.1.

Table A.3: Monte Carlo simulation for an exactly sparse parameter vector with $p = 220$ and $n = 200$.